

On Kernelization and Approximation for the Vector Connectivity Problem

Stefan Kratsch and Manuel Sorge

Technical University Berlin, Germany,
`{stefan.kratsch,manuel.sorge}@tu-berlin.de`

Abstract. In the VECTOR CONNECTIVITY problem we are given an undirected graph $G = (V, E)$, a demand function $\phi: V \rightarrow \{0, \dots, d\}$, and an integer k . The question is whether there exists a set S of at most k vertices such that every vertex $v \in V \setminus S$ has at least $\phi(v)$ vertex-disjoint paths to S ; this abstractly captures questions about placing servers or warehouses relative to demands. The problem is NP-hard already for instances with $d = 4$ (Cicalese et al., arXiv '14), admits a log-factor approximation (Boros et al., Networks '14), and is fixed-parameter tractable in terms of k (Lokshtanov, unpublished '14).

We prove several results regarding kernelization and approximation for VECTOR CONNECTIVITY and the variant VECTOR d -CONNECTIVITY where the upper bound d on demands is a fixed constant. For VECTOR d -CONNECTIVITY we give a factor d -approximation algorithm and construct a vertex-linear kernelization, i.e., an efficient reduction to an equivalent instance with $f(d)k = \mathcal{O}(k)$ vertices. For VECTOR CONNECTIVITY we have a factor opt -approximation and we can show that it has no kernelization to size polynomial in k or even $k + d$ unless $\text{NP} \subseteq \text{coNP/poly}$, making $f(d) \text{poly}(k)$ optimal for VECTOR d -CONNECTIVITY. Finally, we provide a write-up for fixed-parameter tractability of VECTOR CONNECTIVITY(k) by giving an alternative FPT algorithm based on matroid intersection.

1 Introduction

In the VECTOR CONNECTIVITY problem we are given an undirected graph $G = (V, E)$, a demand function $\phi: V \rightarrow \{0, \dots, d\}$, and an integer $k \in \mathbb{N}$. The question is whether there exists a set S of at most k vertices of G such that every vertex $v \in V$ is either in S or has at least $\phi(v)$ vertex-disjoint paths to vertices in S (where the paths may share the vertex v itself).

VECTOR CONNECTIVITY

Instance: A graph $G = (V, E)$, a function $\phi: V \rightarrow \{0, \dots, d\}$, $k \in \mathbb{N}$.

Question: Is there a set S of at most k vertices such that each vertex $v \in V \setminus S$ has $\phi(v)$ vertex-disjoint paths with endpoints in S ?

The value $\phi(v)$ is also called the *demand* of vertex v . We call $S \subseteq V$ a *vector connectivity set* for (G, ϕ) , if it fulfills the requirements above. We do not formally distinguish decision and optimization version; k is not needed for the latter.

For intuition about the problem formulation and applications one may imagine a logistical problem about placing warehouses to service locations on a map (or servers in a network): Each location has a particular demand, which could capture volume and redundancy requirements or level of importance. Demand can be satisfied by placing a warehouse at the location or by ensuring that there are enough disjoint paths from the location to warehouses; both can be seen as guaranteeing sufficient throughput or ensuring access that is failsafe up to demand minus one interruptions in connections. In this way, VECTOR CONNECTIVITY can also be seen as a variant of the FACILITY LOCATION problem [6] in which the costs of serving demand are negligible, but instead redundancy is required.

Related work. The study of the VECTOR CONNECTIVITY problem was initiated recently by Boros et al. [2] who gave polynomial-time algorithms for trees, cographs, and split graphs. Moreover, they obtained an $\ln n + 2$ -factor approximation algorithm for the general case. More recently, Cicalese et al. [3] continued the study of VECTOR CONNECTIVITY and amongst other results proved that it is APX-hard (and NP-hard) on general graphs, even when all demands are upper bounded by four. In a related talk during a recent Dagstuhl seminar¹ Milanič asked whether VECTOR CONNECTIVITY is fixed-parameter tractable with respect to the maximum solution size k . This was answered affirmatively by Lokshtanov.²

Our results. We obtain results regarding kernelization and approximation for VECTOR CONNECTIVITY and VECTOR d -CONNECTIVITY where the maximum demand d is a fixed constant. We also provide a self-contained write-up for fixed-parameter tractability of VECTOR CONNECTIVITY with parameter k (Sec. 5); it is different from Lokshtanov’s approach and instead relies on a matroid intersection algorithm of Marx [11].

Our analysis of the problem starts with a new data reduction rule stating that we can safely “forget” the demand of $r := \phi(v)$ at a vertex v if v has vertex-disjoint paths to r vertices each of demand at least r (Sec. 3). After exhaustive application of the rule, all remaining vertices of demand r must have cuts of size at most $r - 1$ separating them from other vertices of demand at least r . By analyzing these cuts we then show that any yes-instance of VECTOR d -CONNECTIVITY can have at most $d^2 k$ vertices with nonzero demand; the corresponding bound for VECTOR d -CONNECTIVITY is $k^3 + k$. Both bounds also hold when replacing k by the optimum cost opt . This directly would yield factor d^2 and factor $\text{opt}^2 + 1$ approximation algorithms. We improve upon this in Sec. 4 by giving a variant of the reduction rule that works correctly relative to any partial solution S_0 , which can then be applied in each round of our approximation algorithm. The algorithm follows the local-ratio paradigm and, surprisingly perhaps, proceeds by always selecting a vertex of *lowest* demand (plus its cut). We thus obtain ratios of d and opt respectively, i.e., the returned solution is of size at most $d \cdot \text{opt}$ for VECTOR d -CONNECTIVITY and at most opt^2 for VECTOR CONNECTIVITY.

¹ Dagstuhl Seminar 14071 on “Graph Modification Problems.”

² Unpublished result.

Regarding kernelization we show in Sec. 7 that there is no kernel with size polynomial in k or even $k+d$ for VECTOR CONNECTIVITY unless $\text{NP} \subseteq \text{coNP}/\text{poly}$ (and the polynomial hierarchy collapses); our proof also implies that the problem is WK[1]-hard (cf. [7]). Nevertheless, when d is a fixed constant, we prove that a vertex-linear kernelization is possible. A non-constructive proof of this can be pieced together from recent work on meta kernelization on sparse graph classes (see below). Instead we give a constructive algorithm building on an explicit (though technical) description of what constitutes equivalent subproblems. We also have a direct proof for the number of such subproblems in an instance with parameter k rather than relying on a known argument for bounding the number of connected subgraphs of bounded size and bounded neighborhood size (via the two-families theorem of Bollobas, cf. [8]). Our kernelization is presented in Sec. 6. The bound of $f(d)k = \mathcal{O}(k)$ vertices is optimal in the sense that the lower bound for parameter $d+k$ rules out total size $\text{poly}(k+d)$, i.e. we need to allow superpolynomial dependence on d .

A non-constructive kernelization argument. The mentioned results on meta kernelization for problems on sparse graph classes mostly rely on the notion of a protrusion, i.e., an induced subgraph of bounded treewidth such that only a bounded number of its vertices have neighbors in the rest of the graph (the *boundary*). Under appropriate assumptions there are general results that allow the replacements of protrusions by equivalent ones of bounded size, which yields kernelization results assuming that the graph can be decomposed into a bounded number of protrusions. Intuitively, having small boundary size limits the interaction of a protrusion with the remaining graph. The assumption of bounded treewidth ensures fast algorithms for solving subproblems on the protrusion and also leads to fairly general arguments for obtaining small equivalent replacements. Note that for VECTOR CONNECTIVITY there is no reason to assume small treewidth and we also do not restrict the input graphs. Vertex-linear kernels for problems on general graphs are much more rare than the wealth of such kernels on sparse input graphs.

Arguably, the most crucial properties of a protrusion are the small boundary and the fact that we can efficiently compute subproblems on the protrusion; in principle, we do not need bounded treewidth. (Intuitively, we essentially want to know the best solution value for each choice of interaction of a global solution with the boundary of the protrusion.) Thus, it seems worthwhile and natural to define a relaxed variant of protrusions by insisting on a small boundary and efficient algorithms for solving subproblems rather than demanding bounded treewidth. Fomin et al. [5] follow this approach for problems related to picking a certain subset of vertices like DOMINATING SET: Their relaxed definition of a r -DS-protrusion requires a boundary of size at most r and the existence of a solution of size at most r for the protrusion itself; the latter part implies efficient algorithms since we can afford to simply try all $r = \mathcal{O}(1)$ sized vertex subsets. Fomin et al. also derive a general protrusion replacement routine for problems that have finite integer index (a common assumption for meta kernelization, see, e.g., Fomin et al. [5]) and are monotone when provided also with a sufficiently

fast algorithm, like the one implied by having a solution of size at most r for the protrusion. Fomin et al. [5] remark that the procedure is not constructive since it assumes hard-wiring an appropriate set of representative graphs, whose existence is implied by being finite integer index.

From previous work of Cicalese [3] it is known that VECTOR CONNECTIVITY is an implicit hitting set problem where the set family consists of all connected subgraphs with neighborhood size lower than the largest demand in the set; the family is exponential size, but we get size roughly $\mathcal{O}(n^d)$ when demands are at most d . The procedure of Fomin et al. [5] can be applied to minimal sets in this family and will shrink them to some size bounded by an unknown function in d , say $h(d)$. Then one can apply the two-families theorem of Bollobas to prove that each vertex is contained in at most $\binom{h(d)+d}{d}$ such sets. Because the solution must hit all sets with k vertices, a yes-instance can have at most $k \cdot \binom{h(d)+d}{d}$ sets. This argument can be completed to a vertex-linear kernelization.

In comparison, we obtain an explicit upper bound of $d^2 k \cdot 2^{d^3+d}$ for the number of subproblems that need to be replaced, by considering a set family that is different from the implicit hitting set instance (but contains supersets of all those sets). We also have a constructive description of what constitutes equivalent subproblems. This enables us to give a single algorithm that works for all values of d based on maximum flow computations (rather than requiring for each value of d an algorithm with hard-wired representative subproblems).

2 Preliminaries

We consider only undirected, simple graphs $G = (V, E)$ where V is a finite set and $E \subseteq \binom{V}{2}$; we use $V(G) := V$ and $E(G) := E$. (Here, $\binom{V}{2}$ denotes the family of all size-two subsets of V .) The open neighborhood $N_G(v)$ of a vertex $v \in V(G)$ is defined by $N_G(v) := \{u \mid \{u, v\} \in E(G)\}$; the closed neighborhood is $N_G[v] := N_G(v) \cup \{v\}$. For vertex sets $X \subseteq V(G)$ we define $N_G[X] := \bigcup_{v \in X} N_G[v]$ and $N_G(X) := N_G[X] \setminus X$. For $X \subseteq V(G)$ we define the subgraph induced by X , denoted $G[X]$, as the graph $G' = (X, E')$ with $E' := \{\{u, v\} \mid \{u, v\} \in E(G) \wedge u, v \in X\}$. By $G - X$ we mean the graph $G[V(G) \setminus X]$.

Due to the nature of the VECTOR CONNECTIVITY problem we are frequently interested in disjoint paths from a vertex v to some vertex set S , where the paths are vertex-disjoint except for sharing v . The natural counterpart, in the spirit of Menger's theorem, are v, S -separators $C \subseteq V(G) \setminus \{v\}$ such that in $G - C$ no vertex of $S \setminus C$ is reachable from v ; the maximum number of disjoint paths from v to S that may overlap in v equals the minimum size of a v, S -separator. Throughout, by disjoint paths from v to S , or v, S -separator (for any single vertex v and any vertex set S) we mean the mentioned path packings and separators with special role of v . (In a network flow interpretation of the paths between v and S , this essentially corresponds to giving v unbounded capacity, whereas all other vertices have unit capacity.) Several proofs use the function $f: 2^V \rightarrow \mathbb{N}: U \mapsto |N(U)|$, which is well-known to be *submodular*, i.e., for all $X, Y \subseteq V$ it holds that $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$.

So-called *closest sets* will be used frequently; these occur naturally in cut problems but appear to have no generalized name. We define a vertex set C to be *closest to v* if C is the unique v, C -separator of size at most $|C|$, where v, C -separator is in the above sense. As an example, if C is a minimum s, t -vertex cut that, amongst such cuts, has the smallest connected component for s in $G - C$ then C is also closest. The following proposition captures some properties of closest sets, mostly specialized to v, S -separators (see also [9]).

Proposition 1. *Let $G = (V, E)$, let $v \in V$, and let $S \subseteq V \setminus \{v\}$. The following holds.*

1. *If C is a closest v, S -separator and X the connected component of v in $G - C$, then for every set $X' \subseteq X$ with $v \in X' \subsetneq X$ we have $|N(X')| > |N(X)|$.*
2. *The minimum v, S -separator C minimizing the size of the connected component of v in $G - C$ is unique. The set C is also the unique minimum v, S -separator closest to v .*
3. *If C_1 and C_2 are minimum v, S -separators and C_1 is closest to v , then the connected component of v in $G - C_1$ is fully contained in the connected component of v in $G - C_2$.*
4. *If $C \subseteq V \setminus \{v\}$ is closest to v then so is every subset C' of C .*

Proof. Let $G = (V, E)$, let $v \in V$, and let $S \subseteq V \setminus \{v\}$. We give simple proofs for all claims; some of them use submodularity of $f: 2^V \rightarrow \mathbb{N}: U \mapsto |N(U)|$.

1. Note that $N(X) = C$ or else $|N(X)| < |C|$ and $N(X)$ is a v, C -separator, contradicting closeness of C . Let $X' \subseteq X$ with $v \in X' \subsetneq X$. Clearly $X' \cap C \subseteq X \cap C = \emptyset$, making $N(X')$ a v, C -separator. Since $X' \subsetneq X$ and X is connected, it follows that $N(X') \cap X \neq \emptyset$ and, thus, that $N(X') \neq N(X)$. Since C is closest to v , we must have $|N(X')| > |C| = |N(X)|$ or else $N(X')$ would be a v, C -separator of size at most $|C|$ but different from $C = N(X)$.
2. Assume that both C_1 and C_2 are minimum v, S -separators that furthermore minimize the size of the connected component of v in $G - C_i$; let X_i denote those components. Note that $N(X_i) = C_i$ since C_i is a minimum v, S -separator. We have

$$f(X_1) + f(X_2) \geq f(X_1 \cap X_2) + f(X_1 \cup X_2).$$

Clearly, $(X_1 \cup X_2) \cap S = \emptyset$, implying that $N(X_1 \cup X_2)$ is a v, S -separator. It follows that $|N(X_1) \cup N(X_2)| = f(X_1 \cup X_2) \geq f(X_2)$, which implies that $f(X_1) \geq f(X_1 \cap X_2)$. It is easy to see that $N(X_1 \cap X_2)$ is also a v, S -separator of size at most $f(X_1) = |C_1|$ and, if $C_1 \neq C_2$, then $X_1 \neq X_2$ and $X_1 \cap X_2 \subsetneq X_1$; a contradiction to X_1 being a minimum size connected component. Thus, we have $C_1 = C_2$, proving the first part of the claim.

For the second part, assume first that C_3 is a v, C_1 -separator of size at most $|C_1|$ and let X_3 the connected component of v in $G - C_3$. Since every path from v to S contains a vertex of C_1 , each such path must also contain a vertex of C_3 (to separate v from C_1). Thus C_3 is also a v, S -separator, but

then C_3 is also minimum (as $|C_3| \leq |C_1|$). Note that, since C_3 is a v, C_1 -separator we have $X_3 \cap C_1 = \emptyset$. It follows that $X_3 \subseteq X_1$. If $X_3 \subsetneq X_1$ then C_3 would violate the choice of C_1 as minimum v, S -separator with minimum component size for v . But then we must have $X_3 = X_1$ and $C_1 = C_3$, which proves closeness of C_1 to v .

Finally, assume that C_4 is another minimum v, S -separator closest to v ; for uniqueness we want to show $C_1 = C_4$. If X_1 and X_4 are the corresponding connected components, then $C_i = N(X_i)$. By submodularity of f and the same arguments as above we find that $|N(X_1 \cap X_4)| \leq |C_1|$. If $X_1 \not\subseteq X_4$ then $v \in X_1 \cap X_4 \subsetneq X_1$ which, by property 1, implies $|N(X_1 \cap X_4)| > |N(X_1)| = |C_1|$; a contradiction. Else, if $X_1 \subseteq X_4$, then $C_1 = N(X_1)$ is a v, C_4 -separator of size $|C_1| = |C_4|$; this implies $C_1 = C_4$ by closeness of C_4 .

3. Let C_1 and C_2 minimum v, S -separators and let C_1 be closest to v . Let X_i the connected component of v in $G - C_i$. Note that $N(X_i) = C_i$ since C_i is minimum. Assume, for contradiction, that $X_1 \not\subseteq X_2$, implying that $X_1 \cap X_2 \subsetneq X_1$. By property 1 we have $f(X_1 \cap X_2) = |N(X_1 \cap X_2)| > |C_1| = |C_2| = f(X_2)$. Because

$$f(X_1) + f(X_2) \geq f(X_1 \cap X_2) + f(X_1 \cup X_2),$$

we have $|N(X_1 \cup X_2)| < f(X_1) = |C_1| = |C_2|$. However, $N(X_1 \cup X_2)$ is also a v, S -separator; this contradicts C_1 and C_2 being minimum v, S -separators.

4. Let C closest to v and let $C' \subseteq C$. If C' is not closest to v then there is a v, C' -separator C'' of size at most $|C'|$ and with $C'' \neq C'$. Consider $\hat{C} = C'' \cup (C \setminus C')$ and note that $|\hat{C}| \leq |C''| + |C \setminus C'| \leq |C'| + |C \setminus C'| = |C|$. Observe that \hat{C} is a v, C -separator since $C'' \subseteq \hat{C}$ separates v from C' , and $C \setminus C'$ is contained in \hat{C} . Thus, $|\hat{C}| < |C|$ would contradict C being closest to v because it implies that $\hat{C} \neq C$ in addition to \hat{C} being a v, C -separator of size at most $|C|$. Thus, $|\hat{C}| = |C|$ and by closeness of C to v , we have $\hat{C} = C$. The latter can hold only if $C'' \supseteq C'$, because $\hat{C} = C'' \cup (C \setminus C')$, but then $C'' = C'$ because $|C''| \leq |C'|$, contradicting $C'' \neq C'$. Thus C' is indeed closest to v .

This completes the proof. \square

3 Reducing the number of demand vertices

In this section we introduce a reduction rule for VECTOR CONNECTIVITY that reduces the total demand. We prove that the reduction rule does not affect the solution space, which makes it applicable not only for kernelization but also for approximation and other techniques. In Section 6, we will use this rule in our polynomial kernelization for VECTOR d -CONNECTIVITY(k). In the following two sections, as applications of these reduction rules and the insights gained we get approximation algorithms for VECTOR d -CONNECTIVITY and VECTOR CONNECTIVITY, and an alternative FPT algorithm for VECTOR CONNECTIVITY(k) using a result of Marx [11].

Reduction Rule 1. *Let (G, ϕ, k) be an instance of VECTOR CONNECTIVITY. If a vertex $v \in V$ has at least $\phi(v)$ vertex-disjoint paths to vertices different from v with demand at least $\phi(v)$ then set the demand of v to zero.*

We prove that the rule does not affect the space of feasible solutions for any instance.

Lemma 1. *Let (G, ϕ, k) be an instance of VECTOR CONNECTIVITY and let (G, ϕ', k) be the instance obtained via a single application of Rule 1. For every $S \subseteq V(G)$ it holds that S is a solution for (G, ϕ, k) if and only if S is a solution for (G, ϕ', k) .*

Proof. Let v denote the vertex whose demand was set to zero by the reduction rule and define $r := \phi(v)$. Clearly, $\phi(u) = \phi'(u)$ for all vertices $u \in V(G) \setminus \{v\}$, and $\phi'(v) = 0$. It suffices to show that if S fulfills demands according to ϕ' then S fulfills also demands according to ϕ since $\phi(u) \geq \phi'(u)$ for all $u \in V(G)$. This in turn comes down to proving that S fulfills the demand of r at v assuming that it fulfills demands according to ϕ' . If $v \in S$ then the demand at v is trivially fulfilled so henceforth assume that $v \notin S$.

Let w_1, \dots, w_r denote vertices different from v with demand each at least r such that there exist r vertex-disjoint paths from v to $\{w_1, \dots, w_r\}$, i.e., a single path to each w_i . Existence of such vertices is required for the application of the rule.

Assume for contradiction that S does not satisfy the demand of r at v (recall that $v \notin S$, by assumption), i.e., that there are no r vertex-disjoint paths from v to S that overlap only in v . It follows directly that there is a v, S -separator C of size at most $r - 1$. (Recall that C may contain vertices of S but not the vertex v .) Let R denote the connected component of v in $G - C$, then the following holds for each vertex $w_i \in \{w_1, \dots, w_r\}$:

1. If $w_i \in S$ then $w_i \notin R$: Otherwise, we would have $S \cap R \supseteq \{w_i\} \neq \emptyset$ contradicting the fact that v can reach no vertex of S in $G - C$.
2. If $w_i \notin S$ then $w_i \notin R$: Since S fulfills demands according to ϕ' there must be at least r vertex-disjoint paths from w_i to S that overlap only in w_i . However, since $w_i \in R$ the set C is also a w_i, S -separator; a contradiction since C has size less than r .

Thus, no vertex from w_1, \dots, w_r is contained in R . This, however, implies that C separates v from $\{w_1, \dots, w_r\}$, contradicting the fact that there are r vertex-disjoint paths from v to $\{w_1, \dots, w_r\}$ that overlap only in v . It follows that no such v, S -separator C can exist, and, hence, that there are at least $r = \phi(v)$ vertex-disjoint paths from v to S , as claimed. Thus, S fulfills the demand of r at v and hence all demand according to ϕ . (Recall that the converse is trivial since $\phi(u) \geq \phi'(u)$ for all vertices $u \in V(G)$.) \square

We have established that applications of Rule 1 do not affect the solution space of an instance while reducing the number of vertices with nonzero demand.

Lemma 2. *Rule 1 can be exhaustively applied in polynomial time.*

Proof. To check whether the rule applies to some vertex v with demand r it suffices to perform one maximum flow computation for source v (with unbounded capacity) and using all vertices with demand at least r as sinks (with capacity one). Finding a vertex suitable for applying Rule 1 thus takes only polynomial time and each application reduces the number of nonzero demand vertices decreases by one limiting the number of iterations to $|V(G)|$. \square

To analyze the impact of Rule 1 we will now bound the number of nonzero demand vertices in an exhaustively reduced instance in terms of the optimum solution size opt and the maximum demand d . To this end, we require the following technical lemma about the structure of reduced instances as well as some notation. If (G, ϕ, k) is reduced according to Rule 1 then for each vertex v with demand $r = \phi(v) \geq 1$ there is a cut set C of size at most $r - 1$ that separates v from all other vertices with demand at least r . We fix for each vertex v with demand at least one a vertex set C , denoted $C(v)$, by picking the unique closest minimum v, D_v -separator where $D_v = \{u \in V \setminus \{v\} \mid \phi(u) \geq \phi(v)\}$. Furthermore, for such vertices v , let $R(v)$ denote the connected component of v in $G - C(v)$.

Intuitively, any solution S must intersect every set $R(v)$ since $|C(v)| < \phi(v)$. The following lemma shows implicitly that Rule 1 limits the amount of overlap of sets $R(v)$.

Lemma 3. *Let (G, ϕ, k) be reduced under Rule 1. Let $u, v \in V(G)$ be distinct vertices with $\phi(u) = \phi(v) \geq 1$. If $R(u) \cap R(v) \neq \emptyset$ then $u \in C(v)$ or $v \in C(u)$.*

Proof. Assume for contradiction that we have u, v with $\phi(u) = \phi(v) \geq 1$ and with $R(u) \cap R(v) \neq \emptyset$ and $u \notin C(v)$ and $v \notin C(u)$. We will show that this implies that at least one of $C(u)$ and $C(v)$ is not a closest minimum cut, giving a contradiction. By definition of cuts $C(u)$ and $C(v)$ as separating u resp. v from all other vertices of at least the same demand, we have $u \notin R(v)$ and $v \notin R(u)$; furthermore $u \notin C(u)$ and $v \notin C(v)$, by definition.

Let $C = C(u) \cup C(v)$ and note that $u, v \notin C$. Let I, J denote the connected components of u, v in $G - C$. Note that $I \subseteq R(u)$ since $C \supseteq C(u)$ and, thus, $v \notin I$. Similarly we have $u \notin J$ and thus I and J are two different connected components in $G - C$. As the next step, we show that

$$|C(u)| + |C(v)| \geq |N(I)| + |N(J)|. \quad (1)$$

To this end, let us first note that $N(I) \cup N(J) \subseteq C = C(u) \cup C(v)$ by definition of I and J as connected components of $G - C$. Thus, every vertex p that appears in exactly one of $N(I)$ and $N(J)$ contributes value one to the right-hand side of (1) and at least value one to the left-hand side since it must be contained in $C(u) \cup C(v)$. Now, for vertices $p \in N(I) \cap N(J)$ we see that they contribute value two to the right-hand side of (1). Note that each such vertex is contained in a path from u to v whose other interior vertices are disjoint from $C = C(u) \cup C(v)$. Thus, p must be contained in both $C(u)$ and $C(v)$ since otherwise the

corresponding set would fail to separate u from v (or vice versa), which is required since $\phi(u) = \phi(v)$. Therefore, if any vertex contributes a value of two to the right-hand side, then it also contributes two to the left-hand side. This establishes Equation (1).

Now, from (1) we immediately get that at least one of $|N(I)| \leq |C(u)|$ or $|N(J)| \leq |C(v)|$ must hold. W.l.o.g., let $|N(I)| \leq |C(u)|$. Recall that $u \in I$. Furthermore, we can see that $I \subsetneq R(u)$ by using the fact that $R(u) \cap R(v) \neq \emptyset$: Let $q \in R(u) \cap R(v) \subseteq R(u)$. If $q \in I$ then u can reach q in $G - C$ but, in particular, also in $G - C(v)$. By definition of $R(v)$ and using $q \in R(v)$, we know that v can reach q in $G - C(v)$, implying that there is a path from u to v in $G - C(v)$ (since there is a walk through q), violating the fact that $C(v)$ separates v from u (amongst others). Thus $q \notin I$ and since $I \subseteq R(u)$ follows from $C \supseteq C(u)$, we get that $I \subsetneq R(u)$. Since $|N(I)| \leq |C(u)|$ we find that $N(I)$ is of at most the same size as $C(u)$ but with a smaller connected component I for u , contradicting the fact that $C(u)$ is the unique minimum closest set that separates u from all other vertices of demand at least $\phi(u)$. This completes the proof of the lemma. \square

Now, we can give the promised bound on the number of nonzero demand vertices.

Lemma 4. *Let (G, ϕ, k) be an instance of VECTOR CONNECTIVITY that is reduced according to Rule 1 and let opt denote the minimum size of feasible solutions $S \subseteq V$ for this instance. Then there are at most $d^2 \text{opt}$ nonzero demand vertices in G .*

Proof. For analysis, let $S \subseteq V$ denote any feasible solution of size opt , i.e., such that every v with $\phi(v) \geq 1$ has $v \in S$ or there are $\phi(v)$ vertex-disjoint paths from v to S that overlap only in v . We will prove that for all $r \in \{1, \dots, d\}$ there are at most $2r - 1$ vertices of demand r in G (according to ϕ). Fix some $r \in \{1, \dots, d\}$ and let D_r denote the set of vertices with demand exactly r . For each $v \in D_r$ the solution S must contain at least one vertex of $R(v)$ since $C(v) = N(R(v))$ has size at most $r - 1$. (Otherwise, $C(v)$ would be a v, S -separator of size less than r .) Fix some vertex $p \in S$ and let v_1, \dots, v_ℓ denote all vertices of demand r that have $p \in R(v_i)$. We will prove that $\ell \leq (2r - 1)$ and $|D_r| \leq \text{opt}(2r - 1)$.

At most $r - 1$ vertices are contained in $C(v_i)$ for every $i \in \{1, \dots, \ell\}$. Thus, on the one hand the total size $\sum |C(v_i)|$ of these sets is at most $(r - 1)\ell$. On the other hand, for every pair v_i, v_j with $1 \leq i < j \leq \ell$ we know that $v_i \in C(v_j)$ or $v_j \in C(v_i)$ by Lemma 3, since $R(v_i) \cap R(v_j) \supseteq \{p\} \neq \emptyset$. Thus, every pair contributes at least a value of one to the total size of the sets $C(v_i)$. (To see that different pairs have disjoint contributions note that, e.g., $v_i \in C(v_j)$ uniquely defines pair v_i, v_j .) We get the following inequality:

$$(r - 1)\ell \geq \sum_{i=1}^{\ell} |C(v_i)| \geq \binom{\ell}{2}.$$

Thus, $\frac{1}{2}\ell(\ell-1) \leq (r-1)\ell$, implying that $\ell \leq 2r-1$. Since there are exactly opt choices for $p \in S$ and every set $R(v)$ for $v \in D_r$ must be intersected by S , we get an upper bound of

$$\text{opt} \cdot \ell \leq \text{opt}(2r-1)$$

for the size of D_r . If we sum this over all choices of $r \in \{1, \dots, d\}$ we get an upper bound of

$$\sum_{r=1}^d \text{opt}(2r-1) = \text{opt} \sum_{r=1}^d 2r-1 = \text{opt} \cdot d^2$$

for the number of vertices with nonzero demand. This completes the proof. \square

Lemma 4 directly implies reduction rules for VECTOR d -CONNECTIVITY(k) and VECTOR CONNECTIVITY(k): For the former, if there are more than d^2k vertices then opt must exceed k and we can safely reject the instance. For the latter, there can be at most k vertices of demand greater than k since those must be in the solution. Additionally, if $\text{opt} \leq k$ then there are at most $d^2\text{opt} \leq k^3$ vertices of demand at most $d = k$, for a total of $k^3 + k$.

We only spell out the rule for VECTOR d -CONNECTIVITY(k) because it will be used in our kernelization. The bound of $k^3 + k$ for VECTOR CONNECTIVITY(k) will be used for the FPT-algorithm in Section 5.

Reduction Rule 2. *Let (G, ϕ, k) be reduced according to Rule 1, with $\phi: V(G) \rightarrow \{0, \dots, d\}$. If there are more than d^2k vertices of nonzero demand return NO.*

4 Approximation algorithm

In this section we discuss the approximability of VECTOR d -CONNECTIVITY. We know from Lemma 4 that the number of vertices with nonzero demand is at most $d^2\text{opt}$ where opt denotes the minimum size solution for the instance in question. This directly implies a factor d^2 approximation because taking all nonzero demand vertices constitutes a feasible solution. We now show that we can improve on this and develop a factor d approximation for VECTOR d -CONNECTIVITY.

The approximation algorithm will work as follows: We maintain a partial solution $S_0 \subseteq V$, which is initially empty. In each round, we will add at most d vertices to S_0 and show that this always brings us at least one step closer to a solution, i.e., the number of additional vertices that need to be added to S_0 shrinks by at least one. To achieve this, we need to update Rule 1 to take the partial solution S_0 into account.

Reduction Rule 3. *Let (G, ϕ, k) be an instance of VECTOR CONNECTIVITY and let $S_0 \subseteq V(G)$. If there is a vertex v with non-zero demand and a vertex set W not containing v such that each vertex in W has demand at least $\phi(v)$ and v has at least $\phi(v)$ vertex-disjoint paths to $S_0 \cup W$, then set the demand of v to zero. Similarly, if $v \in S_0$ then also set its demand to zero.*

Intuitively, vertices in S_0 get the same status as vertices with demand at least $\phi(v)$ for applying the reduction argument. The proof of correctness now has to take into account that we seek a solution that includes S_0 but the argument stays essentially the same.

Lemma 5. *Let (G, ϕ, k) be an instance of VECTOR CONNECTIVITY, let $S_0 \subseteq V(G)$, and let (G, ϕ', k) be the instance obtained via a single application of Rule 1. For every $S \subseteq V(G)$ it holds that $S \cup S_0$ is a solution for (G, ϕ, k) if and only if $S \cup S_0$ is a solution for (G, ϕ', k) .*

Proof. Let v denote the vertex whose demand was set to zero by the reduction rule and define $r := \phi(v)$. Clearly, $\phi(u) = \phi'(u)$ for all vertices $u \in V(G) \setminus \{v\}$, and $\phi'(v) = 0$. It suffices to show that if $S \cup S_0$ fulfills demands according to ϕ' then $S \cup S_0$ fulfills also demands according to ϕ since $\phi(u) \geq \phi'(u)$ for all $u \in V(G)$. This in turn comes down to proving that S_0 fulfills the demand of r at v assuming that it fulfills demands according to ϕ' . If $v \in S \cup S_0$ then the demand at v is trivially fulfilled; this holds for all S when $v \in S_0$. Thus, we assume henceforth that $v \notin S \cup S_0$. (In particular, the case that $v \in S_0$ is done.)

Let w_1, \dots, w_r denote the r vertices to which we have assumed disjoint paths from v to exist. Each of those vertices is in S_0 or it has demand at least r . Existence of these paths is required to apply the reduction rule when $v \notin S_0$.

Assume for the sake of contradiction that $S \cup S_0$ does not satisfy the demand of r at v (recall that $v \notin S \cup S_0$, by assumption). That is, there are fewer than r vertex-disjoint paths from v to $S \cup S_0$ that overlap only in v . It follows directly that there is a $v, S \cup S_0$ -separator C of size at most $r - 1$. (Recall that C may contain vertices of $S \cup S_0$ but not the vertex v .) Let R denote the connected component of v in $G - C$. Then the following holds for each vertex $w_i \in \{w_1, \dots, w_r\}$:

1. If $w_i \in S \cup S_0$ then $w_i \notin R$: Otherwise, we would have $S \cap R \supseteq \{w_i\} \neq \emptyset$ contradicting the fact that v can reach no vertex of $S \cup S_0$ in $G - C$.
2. If $w_i \notin S \cup S_0$ then $w_i \notin R$: Since $S \cup S_0$ fulfills demands according to ϕ' , and $w_i \notin S \cup S_0$ there must be at least r vertex-disjoint paths from w_i to $S \cup S_0$ that overlap only in w_i . However, if $w_i \in R$, then the set C is also a $w_i, S \cup S_0$ -separator; a contradiction since C has size less than r .

Thus, no vertex from w_1, \dots, w_r is contained in R . This, however, implies that C separates v from $\{w_1, \dots, w_r\}$, contradicting the fact that there are r vertex-disjoint paths from v to $\{w_1, \dots, w_r\}$ that overlap only in v . It follows that no such $v, S \cup S_0$ -separator C can exist, and, hence, that there are at least $r = \phi(v)$ vertex-disjoint paths from v to $S \cup S_0$, as claimed. Thus, $S \cup S_0$ fulfills the demand of r at v and hence all demand according to ϕ . (Recall that the converse is trivial since $\phi(u) \geq \phi'(u)$ for all vertices $u \in V(G)$.) \square

It follows, that we can safely apply Rule 3, as a variant of Rule 1, in the presence of a partial solution S_0 . It is easy to see that also Rule 3 can be applied

exhaustively in polynomial time because testing for any vertex v is a single two-way min-cut computation and each successful application lowers the number of nonzero demand vertices by one.

We now describe our approximation algorithm. The algorithm maintains an instance (G, ϕ) , a set $S_0 \subseteq V(G)$, and an integer $\ell \in \mathbb{N}$. Given an instance (G, ϕ) the algorithm proceeds in rounds to build S_0 , which will eventually be a complete (approximate) solution. We start with $S_0 = \emptyset$ and $\ell = 0$. In any single round, for given (G, ϕ) , set S_0 , and integer ℓ the algorithm proceeds as follows:

1. Exhaustively apply Rule 3 to (G, ϕ) and S_0 , possibly changing ϕ .
2. If S_0 satisfies all demands of (G, ϕ) then return S_0 as a solution (and stop).
3. Otherwise, pick a vertex $v \in V(G)$ of *minimum nonzero demand*. Because we have exhaustively applied Rule 3 there must be a set C of less than $\phi(v) \leq d$ vertices that separates v from S_0 and all vertices of demand at least $\phi(v)$. Add $\{v\} \cup C$ to S_0 and increase ℓ by one. Note that we add at most $\phi(v) \leq d$ vertices to S_0 because $|C| < \phi(v)$.
4. Repeat, i.e., start over with Step 1.

We claim that the algorithm preserves the following invariant.

Invariant 1. *There exists a set S_1 of at most $\text{opt} - \ell$ vertices such that $S_0 \cup S_1$ is a feasible solution for (G, ϕ) .*

We note that the function ϕ may be changed throughout the algorithm, due to applications of Rule 3. Observe that the invariant holds trivially in the beginning, as then $S_0 = \emptyset$ and $\ell = 0$. We now prove that each round of our algorithm preserves the invariant.

Lemma 6. *Each round of the algorithm above preserves Invariant 1.*

Proof. Clearly, by Lemma 5, the invariant is preserved in Step 1 because the sets S that extend S_0 to a solution stay the same. If the algorithm terminates in Step 2 then no more changes are made to S_0 or ℓ so the invariant still holds. It remains to discuss the interesting case that Step 3 happens and we add $\{v\} \cup C$ to S_0 and increase ℓ by one.

For ease of discussion let S'_0 and ℓ' denote S_0 and ℓ from before Step 3. Similarly, fix a set S'_1 of at most $\text{opt} - \ell'$ vertices such that $S'_0 \cup S'_1$ is a feasible solution, as promised by the invariant. We will show that there is a set S_1 of at most $\text{opt} - \ell = \text{opt} - \ell' - 1$ vertices that extends $S_0 = S'_0 \cup \{v\} \cup C$ to a feasible solution.

Let R the connected component of v in $G - C$ and recall that C separates v from all vertices in S_0 and all other vertices of demand at least $\phi(v)$. Because we picked v with minimum nonzero demand, C must in fact separate v from all other nonzero demand vertices. Thus, since Rule 3 has been applied exhaustively, in R there is no vertex of S_0 and no other nonzero demand vertex. The former implies, because $N(R) \subseteq C$ and $|C| < \phi(v)$ that S'_1 must contain at least one vertex of R , say $p \in S'_1 \cap R$. (The latter will be used in a moment.)

We set $S_1 = S'_1 \setminus \{p\}$, noting $|S_1| = |S'_1| - 1$, and claim that $S_0 \cup S_1$ is a feasible solution; this would establish that Invariant 1 holds after Step 3. Let us consider an arbitrary nonzero demand vertex w and check that its demand is satisfied by $S_0 \cup S_1$.

- If $w \in C$ then $w \in S_0 \subseteq S_0 \cup S_1$ and its demand is trivially satisfied.
- If $w \in V \setminus (R \cup C)$ then $w \in S_0 \cup S_1$ if and only if $w \in S'_0 \cup S'_1$ because all changes to these sets are in $R \cup C$. If $w \notin S_0 \cup S_1$ then also $w \notin S'_0 \cup S'_1$ and there must be $r = \phi(w)$ vertex disjoint paths from w to $S'_0 \cup S'_1$, say P'_1, \dots, P'_r . We change the paths to end in $S_0 \cup S_1$: All paths that intersect C can be shortened to end in C . Afterwards, no paths ends in p because it would have to pass C first. Thus all obtained paths, say P_1, \dots, P_r go from w to $S_0 \cup S_1$, and they are vertex-disjoint because they are subpaths of the vertex disjoint paths P'_1, \dots, P'_r (apart from sharing w , of course).
- Finally, if $w \in R$ then we recall that R contains no other nonzero demand vertices except for v . Thus $w = v$ and its demand is fulfilled by $v \in S_0 \subseteq S_0 \cup S_1$.

We find that all steps of our algorithm maintain the invariant, as claimed. \square

Now we can wrap up the section.

Theorem 1. *The VECTOR d -CONNECTIVITY problem admits a polynomial-time factor d approximation.*

Proof. The algorithm works as outlined previously in this section. Given an instance (G, ϕ) of VECTOR d -CONNECTIVITY we start with $S_0 = \emptyset$ and $\ell = 0$ and run the algorithm. We recall that these values of S_0 and ℓ fulfill Invariant 1, recalling that opt denotes the optimum value for vector connectivity sets for (G, ϕ) . In each round, the algorithm adds at most d vertices to S_0 , increases ℓ by one, and always preserves the invariant. Thus, latest when $\ell = \text{opt}$ after opt rounds, it must stop in Step 2 because the invariant guarantees that some set of at most $0 = \text{opt} - \ell$ further vertices gives a solution together with S_0 , i.e., it must find that S_0 is itself a solution. It then outputs S_0 , which, after at most opt rounds, has size at most $d \cdot \text{opt}$. This proves the claimed ratio.

We had already briefly argued that Rule 3 can be applied exhaustively in polynomial time. Similarly, finding the required cut C for a vertex v of minimum demand is polynomial time, and the same is true for testing whether S_0 satisfies all demands. Thus, we indeed have a polynomial-time algorithm that achieves a factor d approximation for VECTOR d -CONNECTIVITY. \square

We can also derive an approximation algorithm for VECTOR CONNECTIVITY, where there is no fixed upper bound on the maximum demand. To this end, we can rerun the previous algorithm for all “guesses” of $\text{opt}_0 \in \{1, \dots, n\}$. In each run, we start with S_0 containing all vertices of demand greater than the guessed value opt_0 , since those must be contained in every solution of total size at most opt_0 . Then the maximum demand is $d = \text{opt}_0$ and we get a d -approximate set of vertices to add to S_0 to get a feasible solution. When $\text{opt}_0 = \text{opt}$, then opt must

also include the same set S_0 and for the remaining $\text{opt} - |S_0| \leq \text{opt}$ vertices we have a d -approximate extension; we get a solution of total size at most opt^2 .

Corollary 1. *The VECTOR CONNECTIVITY problem admits a polynomial-time approximation algorithm that returns a solution of size at most opt^2 , where opt denotes the optimum solution size for the input.*

5 FPT algorithm for Vector Connectivity(k)

In this section we present a randomized FPT-algorithm for VECTOR CONNECTIVITY(k). (We recall that Lokshtanov [10] announced this to be FPT.) Recall that the reduction rules in Section 3 also allow us to reduce the number of nonzero demand vertices to at most $k^3 + k$ (or safely reject). Based on this we are able to give a randomized algorithm building on a randomized FPT algorithm of Marx [11] for intersection of linear matroids. (The randomization comes from the need to construct a representation for the required matroids.) Concretely, this permits us to search for an independent set of size k in $k^3 + k$ linear matroids over the same ground set, where the independent set corresponds to the desired solution and each single matroid ensures that one demand vertex is satisfied. (A matroid is linear if it contains as independent sets precisely the sets of linearly independent columns of a fixed matrix. For more information see, e.g., Oxley's book [12].)

Theorem 2. *VECTOR CONNECTIVITY(k) is randomized fixed-parameter tractable. The error probability is exponentially small in the input size and the error is limited to false negatives.*

Proof (Sketch). We sketch an alternative proof for fixed-parameter tractability of VECTOR CONNECTIVITY(k). W.l.o.g., input instances (G, ϕ, k) are already reduced to at most $k^3 + k$ nonzero demand vertices (else apply the reduction rules); let $D = \{v \in V(G) \mid \phi(v) \geq 1\}$. Clearly, if the instance is YES then there exist also solutions of size *exactly* k (barring $|V(G)| < k$ which would be trivial).

Algorithm. As a first step, we guess the intersection of a solution S^* of size k with the set D ; there are at most $(k^3 + k)^k$ choices for $S_0 = D \cap S^*$. Note that all vertices of demand exceeding k must be contained in S_0 for S^* to be a solution (we ignore S_0 if this is not true).

For each $v \in D \setminus S_0$, we construct a matroid M_v over $V' = V \setminus D$ as follows.

- Build a graph G_v by first adding to G additional $c - 1$ copies of v , called v_2, \dots, v_c , where $c = \phi(v)$, and use $v_1 := v$ for convenience. Second, add $r = k - c$ universal vertices w_1, \dots, w_r (i.e., neighborhood $V \cup \{v_2, \dots, v_c\}$).
- Let M'_v denote the gammoid on G_v with source set $T = \{v_1, \dots, v_c, w_1, \dots, w_r\}$ and ground set $V' \cup S_0$. Recall that the independent sets of a gammoid are exactly those subsets I of the ground set that have $|I|$ vertex-disjoint paths from the sources to I . It is well known that gammoids are linear matroids and that a representation over a sufficiently large field can be found in randomized polynomial time (cf. Marx [11]).

- Create M_v from M'_v by contracting S_0 , making its ground set V' . If S_0 is independent in M'_v then any I is independent in M_v if and only if $S_0 \cup I$ is independent in M'_v .

Use Marx' algorithm [11] to search for a set I^* of size $k - |S_0|$ that is independent in each matroid M_v for $v \in D \setminus S_0$. If a set I^* is found then test whether $S_0 \cup I^*$ is a vector connectivity set for (G, ϕ, k) by appropriate polynomial-time flow computations. If yes then return the solution $S_0 \cup I^*$. Otherwise, if $S_0 \cup I^*$ is not a solution or if no set I^* was found then try the next set S_0 , or answer NO if no set S_0 is left to check.

It remains to prove that the algorithm is correct and to (briefly) consider the runtime.

Correctness. Clearly, if (G, ϕ, k) is NO then the algorithm will always answer NO as all possible solutions $S_0 \cup I^*$ are tested for feasibility.

Assume now that (G, ϕ, k) is YES, let S^* a solution of size k , and let $S_0 = D \cap S^*$. Note that $S^* \subseteq V' \cup S_0$. Pick any $v \in D \setminus S_0$. It follows that there are $c = \phi(v)$ paths from v to S^* in G that are vertex-disjoint except for v . Thus, in G_v we get c (fully) vertex-disjoint paths from $\{v_1, \dots, v_c\}$ to S^* , by giving each path a private copy of v . We get additional $r = k - c$ paths from $\{w_1, \dots, w_r\}$ to the remaining vertices of S^* since $S^* \subseteq V' \cup S_0 \subseteq N(w_i)$. Thus, the set S^* is independent in each gammoid M'_v . Therefore, in each M'_v also $S_0 \subseteq S^*$ is independent. This implies that in M_v (obtained by contraction of S_0) the set $S^* \setminus S_0$ is independent and has size $k - |S_0|$. Moreover, any I is independent in M_v if and only if $I \cup S_0$ is independent in M'_v . It follows, from the former statement, that Marx' algorithm will find *some* set I of size $k - |S_0|$ that is independent in all matroids M_v for $v \in D \setminus S_0$.

We claim that $I \cup S_0$ is a vector connectivity set for (G, ϕ, k) . Let $v \in D \setminus S_0$. We know that I is independent in M_v and, thus, $S := I \cup S_0$ is independent in M'_v . Thus, in G_v there are $|S| = k$ paths from T to S . This entails $c = \phi(v)$ vertex-disjoint paths from $\{v_1, \dots, v_c\}$ to S that each contain no further vertex of T since $|T| = k$. By construction of G_v , we directly get $\phi(v)$ paths from v to S in G that are vertex-disjoint except for overlap in v . Thus, S satisfies the demand of any $v \in D \setminus S_0$. Since $S \supseteq S_0$, we see that S satisfies all demands. Thus, the algorithm returns a feasible solution, as required.

Runtime. Marx' algorithm for finding a set of size k' that is independent in ℓ matroids has FPT running time with respect to $k' + \ell$. We have $k' \leq k$ and $\ell \leq |D| \leq k^3 + k$ in all iterations of the algorithm and there are at most $(k^3 + k)^k$ iterations. This gives a total time that is FPT with respect to k , completing the proof sketch. \square

6 Vertex-linear kernelization for constant demand

In this section we prove a vertex-linear kernelization for VECTOR d -CONNECTIVITY(k), i.e., with d a problem-specific constant and k the parameter. We recall the problem definition.

VECTOR d -CONNECTIVITY(k)

Instance: A graph $G = (V, E)$, a function $\phi: V \rightarrow \{0, \dots, d\}$, and an integer $k \in \mathbb{N}$.

Parameter: k .

Question: Is there a set S of at most k vertices such that each vertex $v \in V \setminus S$ has $\phi(v)$ vertex-disjoint paths with endpoints in S ?

The starting point for our kernelization are Reduction Rules 1 and 2, and a result of Cicalese et al. [3] that relates vector connectivity sets for (G, ϕ) to hitting sets for a family of connected subgraphs of G : Intuitively, if the neighborhood of $X \subseteq V$ in G is smaller than the largest demand of any $v \in X$, then every solution must select at least one vertex in X to satisfy v (by Menger's Theorem). We begin by introducing notation for such a set family but additionally restrict it to (inclusionwise) *minimal* sets X where the demand of some vertex in X exceeds $|N(X)|$. We then state the result of Cicalese et al. [3] using our notation.

Definition 1 ($\mathcal{X}(G, \phi)$). *Let $G = (V, E)$ and let $\phi: V \rightarrow \mathbb{N}$. The family $\mathcal{X}(G, \phi)$ contains all minimal sets $X \subseteq V$ such that*

1. $G[X]$ is connected and
2. there is a vertex $v \in X$ with $\phi(v) > |N(X)|$.

Using this notation, the result of Cicalese et al. [3] is as follows.

Proposition 2 (adapted from Cicalese et al. [3, Proposition 1]). *Let $G = (V, E)$, let $\phi: V \rightarrow \mathbb{N}$, and let $\mathcal{X} := \mathcal{X}(G, \phi)$. Then every set $S \subseteq V$ is a vector connectivity set for (G, ϕ) if and only if it is a hitting set for \mathcal{X} , i.e., it has a nonempty intersection with each $X \in \mathcal{X}$.*

Proof. Cicalese et al. [3] proved Proposition 2 without the minimality restriction, allowing for a larger family, say $\mathcal{X}^+ \supseteq \mathcal{X}$. Clearly, hitting sets for \mathcal{X}^+ are also hitting sets for \mathcal{X} . Conversely, since $\mathcal{X}^+ \setminus \mathcal{X}$ contains only supersets of sets in \mathcal{X} , hitting sets for \mathcal{X} are also hitting sets for \mathcal{X}^+ . \square

Note that for the general case of VECTOR CONNECTIVITY with unrestricted demands the size of $\mathcal{X}(G, \phi)$ can be exponential in $|V(G)|$; for VECTOR d -CONNECTIVITY there is a straightforward bound of $|\mathcal{X}| = \mathcal{O}(|V(G)|^d)$ since $|N(X)| \leq d - 1$. However, even for VECTOR d -CONNECTIVITY, the sets $X \in \mathcal{X}$ are not necessarily small and, thus, we will not take a hitting set approach for the kernelization; in fact, we will not even materialize the set \mathcal{X} but use it only for analysis.

We will leverage Reduction Rules 1 and 2 throughout this section. Hence, we will, sometimes tacitly, assume that all instances of VECTOR CONNECTIVITY are reduced with respect to these rules.

As a first step, we prove that instances (G, ϕ, k) of VECTOR d -CONNECTIVITY(k) that are reduced under Rule 1 have the property that every set $X \in \mathcal{X}(G, \phi)$ contains at most d^3 vertices with nonzero demand. For ease of presentation we define $D(G, \phi) := \{v \in V(G) \mid \phi(v) \geq 1\}$, and use the shorthand $D = D(G, \phi)$ whenever G and ϕ are clear from context.

Lemma 7. *For all $X \in \mathcal{X}$ we have $|X \cap D| \leq (d-1)d^2 \leq d^3$.*

Proof. Recall the definition of $C(v)$ as the unique closest minimum $v, D'(v)$ -separator, where $D'(v) = \{u \in V \setminus \{v\} \mid \phi(u) \geq \phi(v)\}$, and the definition of $R(v)$ as the connected component of v in $G - C(v)$, from Section 3. Fix some $r \in \{1, \dots, d\}$, define $D_r = \{v \in D \mid \phi(v) = r\}$, and consider the relation of $R(v)$ with X for $v \in D_r \cap X$. Note that $D_r \setminus \{v\} \subseteq D'(v)$.

If $R(v) \subsetneq X$, then this would contradict minimality of X : By reducedness under Rule 1, we have $|C(v)| < \phi(v)$ or else the rule would apply to v . But then $R(v)$ with $N(R(v)) = C(v)$ fulfills the conditions for being in \mathcal{X} , except possibly for minimality. This would prevent $X \supsetneq R(v)$ from being included in \mathcal{X} . Else, if $R(v) = X$, then no further vertex of D_r is in X , since $C(v)$ is also a $v, D_r \setminus \{v\}$ -separator as $D_r \setminus \{v\} \subseteq D'(v)$. In this case we get, $|X \cap D_r| = 1$.

In the remaining case, there is no $v \in D_r \cap X$ with $R(v) \subseteq X$. It follows that for all $v \in D_r \cap X$ we have $R(v) \cap X \neq \emptyset$ but $R(v) \not\subseteq X$. This implies $R(v) \cap N(X) \neq \emptyset$ since both $G[X]$ and $G[R(v)]$ are connected. We will use this fact to bound the number of vertices with demand r in X . Let $w \in N(X)$ and let $W \subseteq D_r \cap X$ contain those vertices v of demand r whose set $R(v)$ contains w ; each $R(v)$ must contain at least one vertex in $N(X)$. Thus, for any two vertices $u, v \in W$ we find that their sets $R(u)$ and $R(v)$ have a nonempty intersection, since they share at least w . We can now repeat the same analysis as used in the proof of Lemma 4 to get that $|W| \leq 2r - 1$. Over all choices of w we get an upper bound of $(d-1)(2r-1)$ vertices of demand r in X .

We showed that for each choice of $r \in \{1, \dots, d\}$ we have at most $(d-1)(2r-1)$ vertices of demand r in X . Summing over all $r \in \{1, \dots, d\}$ this yields an upper bound of $(d-1)d^2 \leq d^3$ for $|X \cap D|$, as claimed. \square

To arrive at our kernelization we will later establish a reduction rule that shrinks connected subgraphs with small boundary and bounded number of demand vertices to constant size. This is akin to blackbox protrusion-based reduction rules, especially as in [5], but we give an explicit algorithm that comes down to elementary two-way flow computations. To get an explicit (linear) bound for the number of subproblems, we introduce a new family \mathcal{Y} with larger but (as we will see) fewer sets, and apply the reduction process to graphs $G[Y]$ with $Y \in \mathcal{Y}$ instead. Alternatively, as pointed out in the introduction, one may use a result of Bollobas for bounding the number of sets in \mathcal{X} once they are small; a caveat is that this bound would depend on the final size of sets in \mathcal{X} , whereas we have a direct and explicit bound for $|\mathcal{Y}|$.

Definition 2 ($\mathcal{Y}(G, \phi, d)$). *Let $G = (V, E)$, let $d \in \mathbb{N}$, and let $\phi: V \rightarrow \{0, \dots, d\}$. The family $\mathcal{Y}(G, \phi, d)$ contains all sets $Y \subseteq V$ with*

1. $G[Y]$ is connected,
2. $|Y \cap D| \leq d^3$, i.e., Y contains at most d^3 vertices v with nonzero demand $\phi(v)$,
3. $|N(Y)| \leq d$, i.e., Y has at most d neighbors, and
4. there is a vertex $v \in Y \cap D$, i.e., $\phi(v) \geq 1$, such that $N(Y)$ is the unique closest minimum $v, D \setminus Y$ -separator.

For (G, ϕ, d) , we relate $\mathcal{X} = \mathcal{X}(G, \phi)$ and $\mathcal{Y} = \mathcal{Y}(G, \phi, d)$ by proving that every set $X \in \mathcal{X}$ is contained in at least one $Y \in \mathcal{Y}$. Intuitively, this proves that all “interesting” parts of the instance are contained in subgraphs $G[Y]$.

Lemma 8. *Let $G = (V, E)$ a graph, $d \in \mathbb{N}$, and $\phi: V \rightarrow \{0, \dots, d\}$. Let $\mathcal{X} := \mathcal{X}(G, \phi)$ and $\mathcal{Y} := \mathcal{Y}(G, \phi, d)$. Then for all $X \in \mathcal{X}$ there exists $Y \in \mathcal{Y}$ with $X \subseteq Y$.*

Proof. Let $X \in \mathcal{X}$ and pick $v \in X$ with $\phi(v) > |N(X)|$. Let $D_1 = D \setminus X$, i.e., those nonzero demand vertices that are not in X . Now, let $Z \subseteq V \setminus \{v\}$ the minimum v, D_1 -separator that is closest to v , and let Y be the connected component of v in $G - Z$; thus $Z = N(Y)$. We claim that $X \subseteq Y$ and $Y \in \mathcal{Y}$.

First, assume for contradiction that $X \not\subseteq Y$. We use submodularity of $f: 2^V \rightarrow \mathbb{N}: U \mapsto |N(U)|$, which implies

$$f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y). \quad (2)$$

Note that $D_1 \cap (X \cup Y) = (D_1 \cap X) \cup (D_1 \cap Y) = \emptyset$ and that $v \in X \cup Y$. It follows that $N(X \cup Y)$ is also a v, D_1 -separator and, using that Z is minimum, we get that $f(X \cup Y) = |N(X \cup Y)| \geq |Z| = |N(Y)| = f(Y)$. Plugging this into (2) we obtain $f(X) \geq f(X \cap Y)$. Note that $v \in X \cap Y$ and $D_1 \cap (X \cap Y) = \emptyset$, implying that $N(X \cap Y)$ is also a v, D_1 -separator of size at most $|N(X)| = f(X)$. From $X \not\subseteq Y$ we get $X \cap Y \subsetneq X$. But then $X \cap Y$ is a proper subset of X containing v and having $|N(X \cap Y)| \leq |N(X)| < \phi(v)$. It follows that the connected component of v in $G[X \cap Y]$ also has neighborhood size (in G) less than $\phi(v)$. This contradicts the fact that X is a minimal set fulfilling the properties of Definition 1, which is required for $X \in \mathcal{X}$. We conclude that, indeed, $X \subseteq Y$.

Second, let us check that Y fulfills the requirements for $Y \in \mathcal{Y} = \mathcal{Y}(G, \phi, d)$ (as in Definition 2). Note that Z separates v from all nonzero demand vertices that are not in X , since $D_1 = D \setminus X$. Thus, every nonzero demand vertex in Y is also contained in X , i.e., $D \cap Y \subseteq D \cap X$, which bounds their number by d^3 using Lemma 7. It also follows that $D \setminus X = D \setminus Y$, since $X \subseteq Y$ implies $D \cap X \subseteq D \cap Y$. Furthermore, $G[Y]$ is connected and $N(Y)$ is a minimum $v, D \setminus Y$ -separator that is closest to v ; note that $D_1 = D \setminus X = D \setminus Y$. Finally, since $N(X)$ is also a v, D_1 -separator and $N(Y)$ is minimum, we conclude that $|N(Y)| \leq |N(X)| < \phi(v) \leq d$. Thus, indeed, $Y \in \mathcal{Y}$ as claimed. \square

We prove that the number of sets $Y \in \mathcal{Y}$ is linear in k for every fixed d . Thus, by later shrinking the size of sets in \mathcal{Y} to some constant we get $\mathcal{O}(k)$ vertices in total over sets $Y \in \mathcal{Y}$.

Lemma 9. *Let (G, ϕ, k) an instance of VECTOR d -CONNECTIVITY(k) with $\phi: V(G) \rightarrow \{0, \dots, d\}$ and let $\mathcal{Y} := \mathcal{Y}(G, \phi, d)$. Then $|\mathcal{Y}| \leq d^2 k \cdot 2^{d^3+d}$.*

Proof. We prove the lemma by giving a branching process that enumerates all sets $Y \in \mathcal{Y}$ within the leaves of a branching tree and by showing that the tree has at most $d^2 k \cdot 2^{d^3+d}$ leaves in which sets Y are found. Given $G = (V, E)$, $\phi: V \rightarrow \{0, \dots, d\}$, and $k \in \mathbb{N}$, the process works as follows. (Recall $D = \{v \in V \mid \phi(v) \geq 1\}$.)

1. As a first step, *branch* on choosing one vertex $v \in D$. Recall that an instance reduced with respect to Rule 2 has $|D| \leq d^2k$. Hence, this branching step incurs a factor of $|D| \leq d^2k$ to the number of leaves and creates one node for each choice.
2. Maintain disjoint sets $D_0, D_1 \subseteq D$, starting with $D_0 := \{v\}$ and $D_1 := \emptyset$, and the minimum v, D_1 -separator Z that is closest to v ; initially $Z = \emptyset$. Throughout, use Y to refer to the connected component of v in $G - Z$.
3. All further *branchings* work as follows: Pick an arbitrary vertex $p \in D \setminus (D_0 \cup D_1)$ that is reachable from v in $G - Z$. *Branch* on either adding this vertex to D_0 or to D_1 , creating a child node for each choice. In the branch where p is added to D_1 update the minimum v, D_1 -separator Z closest to v and update the connected component Y of v in $G - Z$.
4. Terminate a branch if any of the following three *termination conditions* occurs.
 - (a) The size of D_0 exceeds d^3 .
 - (b) The size of Z exceeds d .
 - (c) No vertex of $D \setminus (D_0 \cup D_1)$ is reachable from v in $G - Z$.

We will now analyze this process. First, we show that every set of \mathcal{Y} occurs as the set Y in some leaf node of the process, i.e., a node to which a termination condition applies. Second, we show that the number of such leaf nodes is bounded by $|D| \cdot 2^{d^3+d} \leq d^2k \cdot 2^{d^3+d}$.

Each set of \mathcal{Y} occurs in some leaf. We show that *every* set $Y^* \in \mathcal{Y}$ is found as the set Y of at least one leaf of the branching tree. To this end, fix an arbitrary set $Y^* \in \mathcal{Y}$ and let $Z^* := N_G(Y^*)$. Furthermore, let $D_0^* := Y^* \cap D$. Let $v \in Y^* \cap D$ such that $Z^* = N(Y^*)$ is the unique minimum, closest $v, D \setminus Y^*$ -separator. In the first branching step the process can clearly pick v for its choice of vertex in D ; in this case it continues with $D_0 = \{v\}$, $D_1 = \emptyset$, $Z = \emptyset$, and Y is the connected component of v in G . Consider nodes in the branching process with current sets Y , D_0 , D_1 , and $Z = N(Y)$ fulfilling the *requirements* that

- $Y \supseteq Y^*$ and
- $D_0 \subseteq D_0^*$ and $D_1 \cap D_0^* = \emptyset$.

Among such nodes pick one that is either a leaf or such that neither child node fulfills the requirements. Clearly, the node with $D_0 = \{v\}$, $D_1 = \emptyset$, $Z = \emptyset$, and Y equal to the component of v in G fulfills the requirements, so we can indeed always find such a node by starting at this one and following child nodes fulfilling the requirements until reaching a leaf or until both child nodes do not fulfill the requirements.

Leaf node fulfilling the requirements. If the chosen node is a leaf then one of the three termination conditions must hold. Clearly, we cannot have $|D_0| > d^3$ since that would imply $|D_0^*| > d^3$, violating the definition of \mathcal{Y} and the sets therein. Similarly, we cannot have $|Z| > d$: In this regard, note that $D_1 \cap D_0^* = \emptyset$ implies that $D_1 \subseteq D \setminus D_0^* = D \setminus (D \cap Y^*) = D \setminus Y^*$. It follows directly that Z^* , which separates v from $D \setminus Y^*$, also separates v from D_1 . But then the size of

Z^* is an upper bound for the minimum cut size for v, D_1 -separators and $|Z| > d$ would imply $|Z^*| \geq |Z| > d$, again violating the definition of \mathcal{Y} .

Thus, in case of a leaf node the only remaining option is that no further vertex of $D \setminus (D_0 \cup D_1)$ is reachable from v in $G - Z = G - N(Y)$. Recall that Z is the minimum closest v, D_1 -separator. By termination Z also separates v from $D \setminus (D_0 \cup D_1)$, making it a closest $v, D \setminus D_0$ -separator. Since $D_0 \subseteq D_0^*$, it follows that $D \setminus Y^* = D \setminus D_0^* \subseteq D \setminus D_0$. Thus, the size of the minimum $v, D \setminus Y^*$ -separator Z^* is upper-bounded by $|Z|$ since Z is also a $v, D \setminus Y^*$ -separator. Recall that we have $Y \supseteq Y^*$, which implies that $Z^* = N(Y^*)$ also separates v from $Z = N(Y)$. Now, because Z is closest to v , it is the unique v, Z -separator of size at most $|Z|$, which implies $Z = Z^*$ since we just derived that $|Z^*| \leq |Z|$. Thus, $Y = Y^*$ since both are identified as the connected component of v in $G - Z = G - Z^*$. We conclude that Y^* is equal to Y in the chosen node if it is a leaf.

Internal node fulfilling the requirements. Now, let us consider the case that the chosen node is not a leaf of the branching tree. We want to check that at least one possible branch must lead us to a child node that also fulfills our restrictions; this would contradict our choice of node that is either a leaf or such that neither child node fulfills the requirements, implying that we necessarily pick a leaf node. Thus, for any $Y^* \in \mathcal{Y}$ there is a leaf of the branching tree with $Y = Y^*$. For clarity, in the following discussion we will use Y, D_0 , etc. for the current node and Y', D'_0 , etc. for the considered child node in the branching tree.

Since we are not in a leaf in this case, the process chooses an arbitrary vertex $p \in D \setminus (D_0 \cup D_1)$ that is reachable from v in $G - Z$ to branch on. If $p \in D_0^*$ then the child node corresponding to adding p to D_0 has $D'_0 = D_0 \cup \{p\} \subseteq D_0^*$ and $D'_1 = D_1$. Thus, $Z' = Z$ and, hence, $Y' = Y \supseteq Y^*$. Thus, the child node fulfills all requirements; a contradiction.

Otherwise, if $p \notin D_0^*$, then $p \in D \setminus D_0^* = D \setminus Y^*$. Thus, the child node corresponding to adding p to D_1 has $D'_0 = D_0 \subseteq D_0^*$ and $D'_1 = D_1 \cup \{p\}$, implying that $D'_1 \cap D_0^* = D_1 \cap D_0^* = \emptyset$. For the desired contradiction it remains to prove that $Y' \supseteq Y^*$ since we assumed that neither child fulfills the requirements.

Assume that $Y^* \not\subseteq Y'$. Thus, $Y^* \cap Y' \subsetneq Y^*$. We again use submodularity of the function $f: 2^V \rightarrow \mathbb{N}: U \mapsto |N(U)|$ and get

$$f(Y^*) + f(Y') \geq f(Y^* \cap Y') + f(Y^* \cup Y'). \quad (3)$$

Since $v \in Y^* \cap Y' \subsetneq Y^*$ and $N(Y^*)$ is closest to v it follows that $f(Y^* \cap Y') = |N(Y^* \cap Y')| > |N(Y^*)| = f(Y^*)$, by Proposition 1. Plugging this into (3) yields $f(Y^* \cup Y') < f(Y')$; let us check that this violates the fact that $Z' = N(Y')$ is a minimum v, D'_1 -separator: We have $v \in Y^* \cup Y'$ and

$$D'_1 \cap (Y^* \cup Y') = \underbrace{(D'_1 \cap Y^*)}_{\subseteq D} \cup \underbrace{(D'_1 \cap Y')}_{=\emptyset} = (D'_1 \cap Y^*) \cap D = D'_1 \cap D_0^* = \emptyset.$$

Thus, indeed, we find that $N(Y^* \cup Y')$ is a v, D'_1 -separator and we know from $f(Y^* \cup Y') < f(Y')$ that it is smaller than the assumed minimum v, D'_1 -separator $Z' = N(Y')$; a contradiction. Hence, by our choice of node that fulfills the

requirements and is a leaf or neither child fulfills the requirements, we must obtain a leaf with set Y equal to Y^* .

Number of leaves containing some $Y^* \in \mathcal{Y}$. We have seen that every set $Y^* \in \mathcal{Y}$ is equal to the set Y of some leaf node fulfilling certain requirements. Furthermore, leaves with $|D_0| > d^3$ or $|Z| > d$ were shown not to correspond to any $Y \in \mathcal{Y}$. We will now analyze the number of leaf nodes with $|D_0| \leq d^3$ and $|Z| \leq d$.

Crucially, we show that each branching increases $|D_0| + |Z|$. For adding p to D_0 this is obvious, for adding p to D_1 we prove this next. Concretely, we prove that adding p to D_1 increases the minimum size of v, D_1 -separators by at least one. (This is essentially folklore but we provide a proof for completeness.) Use $D'_1 = D_1 \cup \{p\}$ and let Z and Z' denote minimum v, D_1 - and v, D'_1 -separators closest to v ; use Y and Y' for the components of v in $G - Z$ and $G - Z'$, respectively. We use again the submodular function $f: 2^V \rightarrow \mathbb{N}: U \rightarrow |N_G(U)|$ and obtain

$$f(Y) + f(Y') \geq f(Y \cup Y') + f(Y \cap Y'). \quad (4)$$

Both Z and Z' separate v from D_1 . Thus, $D_1 \cap (Y \cup Y') = \emptyset$ and $N(Y \cup Y')$ is also a v, D_1 -separator, since it creates the component $Y \cup Y'$ for v in $G - N(Y \cup Y')$. Since Z is a minimum v, D_1 -separator we must have $f(Y \cup Y') = |N(Y \cup Y')| \geq |Z| = f(Y)$. Plugging this into (4) yields $f(Y \cap Y') \leq f(Y')$. If $f(Y') = |Z'| \leq |Z| = f(Y)$, i.e., if adding p to D_1 does not increase the minimum size of v, D_1 -separators,³ then $f(Y \cap Y') \leq |Z|$ and $N(Y \cap Y')$ is also a v, D_1 -separator of size at most $|Z|$. However, as $p \notin Y'$, since Z' separates v from $D'_1 = D_1 \cup \{p\}$, the set $Y \cap Y'$ is a strict subset of Y ; this is a contradiction to Z being closest to v . As a consequence, the size of closest, minimum v, D_1 -separators increases whenever we branch into adding a so far reachable vertex to D_1 .

Thus, every child node has $|D'_0| + |Z'| \geq |D_0| + |Z| + 1$ and, clearly, neither value decreases when branching. Thus, the process can only reach leaf nodes with $|D_0| \leq d^3$ and $|Z| \leq d$ via internal nodes where $|D_0| + |Z| \leq d^3 + d$. It follows that the number of such leaf nodes is upper bounded by $|D| \cdot 2^{d^3+d} \leq d^2 k \cdot 2^{d^3+d}$. (Once $|D_0| \geq d^3$ or $|Z| \geq d$ at most one child node can lead to such a leaf, since the other child violates the restriction on $|D_0|$ or $|Z|$; these branches are not counted.) Thus, $|\mathcal{Y}| \leq d^2 k \cdot 2^{d^3+d}$, as claimed. \square

Reducing the size of sets in \mathcal{Y} . In this part, we explain and prove how to reduce the size of sets $Y \in \mathcal{Y}$ through modifications on the graph G . At a high level, this will be achieved by replacing subgraphs $G[Y]$ by “equivalent” subgraphs of bounded size. When this is done, we know that the total number of vertices in sets $Y \in \mathcal{Y}$ is $\mathcal{O}(k)$. Since this part is somewhat technical and long, let us try to illustrate it first.

Consider a set $Y \in \mathcal{Y}$ and its (small) neighborhood $Z := N_G(Y)$. Think of deciding whether (G, ϕ, k) is YES as a game between two players, Alice and

³ These values coincide with $f(Y)$ and $f(Y')$ since we chose $Z = N(Y)$ and $Z' = N(Y')$ as minimum v, D_1 - and v, D'_1 -separators.

Bob. Alice sees only $G[Y \cup Z]$ and wants to satisfy the demands of all vertices in Y , and Bob sees only $G - Y$ and wants to satisfy the demands of the vertices in $V \setminus Y$. To achieve a small solution the players must cooperate and exchange information about paths between vertices in Z , or between Z and vertices of a partial solution, that they can *provide* or that they *require*.

Since our goal is to simplify $G[Y]$, all notation is given using Alice's perspective. Crucially, we know that there are only constantly many nonzero demand vertices in Y , which can be seen to imply that the intersection of optimal solutions with Y is bounded (Lemma 10 below). Thus, Alice can try all partial solutions $S_Y \subseteq Y$ of bounded size and determine what *facilities* each S_Y provides for Bob, and what *requirements* she has on Bob to satisfy her demand vertices using S_Y and further paths through $G - Y$.

Let us be slightly more concrete about facilities and requirements, before making them fully formal. If we fix some partial solution $S_Y \subseteq Y$ then Alice can offer (as facilities) to Bob to connect some subsets of Z to S_Y by disjoint paths in $G[Y \cup Z]$, and to, additionally, provide paths connecting certain sets of vertices in Z . There can be a large number of such options for each S_Y . Similarly, to fulfill demands in Y , Alice may need (as requirements) that Bob can provide paths from certain subsets of Z to a solution and, additionally, paths connecting sets of vertices in Z . (Note that there is some asymmetry here since Bob's part is too large to fully analyze, but this will be no problem.) Fortunately, while there may be many choices for S_Y , and many facilities and requirements relative to a single S_Y , it will turn out that the overall number of things to keep track of is bounded (in d); this will be called the *signature* of $G[Y \cup Z]$. Ultimately, we will be able to replace $G[Y \cup Z]$ by a bounded-size graph with the same signature.

We now make our approach formal. For convenience, let us introduce the following notation. A *separation* of a graph G is a tuple (T, U) of two vertex subsets $T, U \subseteq V(G)$ such that $T \cup U = V(G)$ and there is no edge between $T \setminus U$ and $U \setminus T$ in G . The *order* of a separation (T, U) is $|T \cap U|$. We call a set of paths to be *v -independent* if each pair of paths is vertex-disjoint except for possibly sharing v as an endpoint. For a graph G , a vertex v , an integer i , and two vertex subsets $A, B \subseteq V(G)$ we define a (v, i, A, B) -*constrained path packing* as a set of $i + |A|$ v -independent paths from $A \cup \{v\}$ to B in G . Herein we explicitly allow $v \notin V(G)$ if $i = 0$. If $i = 0$, we simplify the notation and speak of (A, B) -constrained path packings instead. Note that, regardless of whether $v \in V(G)$, the paths saturate each vertex in A . Furthermore, we tacitly assume that, if $A \cap B \neq \emptyset$ then the paths corresponding to $A \cap B$ in the packing are of length zero, that is, they each comprise a single vertex.

Let us now begin with the definition of signatures. In the following, let G and ϕ represent the graph and demand function of an instance of VECTOR d -CONNECTIVITY(k). To decrease the necessary notation we also fix a set $Y \subseteq V(G)$ such that $|Y \cap D| \leq d^3$ and $|N(Y)| \leq d$. (In the kernelization procedure, the role of Y will be assumed by some set in \mathcal{Y} .) We denote the neighborhood $N(Y)$ by Z .

We will first take care of the requirements that Alice has. The facilities will be treated later. As mentioned above, a requirement comprises several sets of paths outside of $G[Y]$ one set of which needs to be provided by Bob (and his part of the solution) in order to satisfy the demand of a vertex $v \in D \cap Y$. To this end, we define a *satisfying connector*. In the following $Y \uplus Z$ denotes the disjoint union of the sets Y and Z .

Definition 3 (Satisfying connector). *Let H be a graph on vertex set $Y \uplus Z$, let $v \in Y$, let v have positive demand d_v , and let $S_Y \subseteq Y$ be a partial solution. A tuple (A, B, C) with $A, B, C \subseteq Z$, pairwise disjoint, is a satisfying connector for v with respect to S_Y in H if either $v \in S_Y$ or there is a $(v, d_v, A, B \cup S_Y)$ -constrained path packing in $H - C$. The set of all satisfying connectors for v with respect to the partial solution S_Y is denoted by $\text{Sat}(H, Z, d_v, S_Y, v)$.*

Now we can formally define the requirement of Alice. Intuitively, Bob should provide paths that “hit” each set of satisfying connectors induced by a vertex with positive demand.

Definition 4 (Requirement). *Let H be a graph on vertex set $Y \uplus Z$, let ϕ be a demand function on Y , and let $S_Y \subseteq Y$ be a partial solution. The requirement $\text{Req}(H, Z, \phi, S_Y)$ is the collection $\{\text{Sat}(H, Z, \phi(v), S_Y, v) \mid v \in D(H, \phi) \cap Y\}$.*

After the definition of signatures we will prove that the demand of each vertex in $D(\phi) \cap Y$ can be met if and only if the requirement of Alice is met by a suitable path packing provided by Bob.

Note that, in order to be able to replace $G[Y]$ by a different graph, we need to know which requirements it imposes for any relevant choice of the partial solution S_Y . Since we are aiming for a polynomial-time kernelization, we also need to be able to compute them in polynomial time. For this, we first bound the size of S_Y .

Lemma 10. *For each vector connectivity set S of (G, ϕ) , there is a vector connectivity set S' such that $|S'| \leq |S|$ and such that $|Y \cap S'| \leq d^3 + d$.*

Proof. Assume that $|Y \cap S| \geq d^3 + d$. Remove all vertices in $(Y \cup Z) \cap S$ from S and add all vertices in $D \cap Y$ as well as all vertices in Z . Denote the resulting vertex set by S' . Recall that $|D \cap Y| \leq d^3$ by definition of Y . Hence, $|Y \cap S'| \leq |D \cap Y| + |N_G(Y)| \leq d^3 + d$ and, thus, $|S'| \leq |S|$. Clearly, S' satisfies all demands of vertices in Y and Z since $(D \cap Y) \cup Z \subseteq S'$. For vertices $v \in D \setminus (Y \cup Z)$, note that at most $|Z|$ paths used for reaching S from v can traverse Z , and all of those can be shortened to end in $Z \subseteq S'$; all other paths avoid Z and thereby $Y \cup Z$, implying that they still end in vertices of $S \setminus (Y \cup Z) = S' \setminus (Y \cup Z)$. \square

We are almost ready to compute the requirements; crucially, we need to check whether there are suitable (v, d, A, B) -constrained path packings in polynomial time.

Lemma 11. *Let G be a graph, $v \in V(G)$, $d \in \mathbb{N}$, and $A, B \subseteq V(G)$. It is possible to check in polynomial time, whether there is a (v, d, A, B) -constrained path packing in G .*

Proof (Sketch). We reduce the task to computing a maximum flow in a modified graph: First, remove each vertex in $A \cap B$ from the graph—we can assume that they represent paths of length zero in the desired path packing. Then, add d copies of v to G , each adjacent to all neighbors of v . Then, attach a new vertex s to each vertex in A and the copies of v , and attach a new vertex t to each vertex in B . It is not hard to check that there are $d + |A \setminus B|$ internally vertex-disjoint paths from s to t in the modified graph if and only if there is a (v, d, A, B) -constrained path packing in the original graph.

Checking whether there are enough internally vertex-disjoint s - t paths can be done using a folklore reduction to maximum flow: Create a flow network by introducing two vertices v_{in} and v_{out} for each vertex $v \in V(G)$, and also an arc $(v_{\text{in}}, v_{\text{out}})$ of capacity one. Then, for each edge $\{u, v\}$, add two arcs $(u_{\text{out}}, v_{\text{in}}), (v_{\text{out}}, u_{\text{in}})$ with capacity infinity. One can check that there is a flow of value $d + |A \setminus B|$ between s_{out} and t_{in} if and only if the desired path packing exists. \square

Lemma 12. *Let $Y \in \mathcal{Y}$, $Z = N(Y)$, and $H = G[Y \cup Z]$. The set $\{\text{Req}(H, Z, \phi|_Y, S_Y) \mid S_Y \subseteq Y \wedge |S_Y| \leq d^3 + d\}$ is computable in polynomial time. Herein, $\phi|_Y$ denotes ϕ restricted to Y .*

Proof. By enumerating all $S_Y \subseteq Y$ of size at most $d^3 + d$ in $n^{\mathcal{O}(d^3)}$ time, i.e. polynomial time, the task reduces to computing $\text{Req}(H, Z, \phi|_Y, S_Y)$ for a given S_Y . To do this, we compute the set of satisfying connectors for each vertex v in $D \cap Y$. This in turn we do by simply iterating over all tuples (A, B, C) with $A, B, C \subseteq Z$, pairwise disjoint, and we check whether it is contained in $\text{Sat}(H, Z, \phi(v), S_Y, v)$. Note that $|Z| \leq d$. Hence, there are at most $2^{\mathcal{O}(d)} \in \mathcal{O}(1)$ different tuples to check. Thus, it remains to check whether in $H - C$ there is a $(v, \phi(v), A, B \cup S_Y)$ -constrained path packing. This can be done using Lemma 11. \square

So far we have only talked about the requirements that Alice has on Bob. Now let us come to the facilities that Alice provides. As mentioned, a facility models the sets of paths inside of $G[Y]$ that Alice, using her part of the solution, provides to Bob so to satisfy the demand of each vertex in $D \setminus Y$. Let us first focus on vertices in $D \setminus (Y \cup Z)$.

Definition 5 (Provided connector). *Let H be a graph on vertex set $Y \uplus Z$, and let $S_Y \subseteq Y$ be a partial solution. A tuple (A, B, C) with $A, B, C \subseteq Z$, pairwise disjoint, is a provided connector of S_Y in H if there is a $(A, B \cup S_Y)$ -constrained path packing in $H - C$.*

We prove below that all demands of vertices in $D \setminus (Y \cup Z)$ can be met if and only if Alice provides suitable path packings to Bob. We take special care of vertices in $D \cap Z$, as they may need multiple paths into $G[Y]$.

Definition 6 (Provided special connector). Let H be a graph on vertex set $Y \uplus Z$ and let $S_Y \subseteq Y$. A tuple (z, i, A, B, C) with $z \in Z$, $A, B, C \subseteq Z \setminus \{z\}$, pairwise disjoint, and $i \in \{0, \dots, d\}$ is a provided special connector of S_Y in H if there is a $(z, i, A, B \cup S_Y)$ -constrained path packing in $H - C$.

We are now ready to give a formal definition of the facilities provided by Alice.

Definition 7 (Facility). Let H be a graph on vertex set $Y \uplus Z$, and let $S_Y \subseteq Y$ be a partial solution. The facility $\text{Fac}(H, Z, S_Y)$ of S_Y in H is the set of all provided connectors and provided special connectors of S_Y .

Similarly to requirements, we need an efficient algorithm for computing the facilities; this basically follows from Lemma 11.

Lemma 13. Let $Y \in \mathcal{Y}$, $Z = N(Y)$, and $H = G[Y \cup Z]$. The set $\{\text{Fac}(H, Z, S_Y) \mid S_Y \subseteq Y \wedge |S_Y| \leq d^3 + d\}$ is computable in polynomial time.

Proof. We first enumerate all $S_Y \subseteq Y$ with $|S_Y| \leq d^3 + d$ in $n^{\mathcal{O}(d^3)}$ time and, for each such S_Y , we compute all provided connectors and provided special connectors. This is done by iterating over all possible tuples (A, B, C) and (z, i, A, B, C) (there are at most $d^2 \cdot 2^{\mathcal{O}(d)} \in \mathcal{O}(1)$ of them) and checking whether they indeed are provided (special) connectors. This is done using Lemma 11. \square

Now we can precisely define the signature of $G[Y \cup Z]$ that we mentioned earlier.

Definition 8 (Signature). Let H be a graph on vertex set $Y \uplus Z$, let ϕ be a demand function on Y , and let $S_Y \subseteq Y$ be a partial solution. The signature of H is the set

$$\text{Sig}(H, Z, \phi) := \{(|S_Y|, \text{Req}(H, Z, \phi, S_Y), \text{Fac}(H, Z, S_Y))\},$$

where S_Y ranges over all $S_Y \subseteq Y$ such that $|S_Y| \leq d^3 + d$.

We show below that we can safely replace $G_Z[Y \cup Z]$ with any graph $G'[Y' \cup Z]$ that has the same signature. Let us now prove that there is a graph $G'[Y' \cup Z]$ of constant size with the same signature.

Lemma 14. There is a polynomial-time algorithm that receives $Y \in \mathcal{Y}$, $Z = N(Y)$, $G[Y \cup Z]$, and $\phi|_Y$ as input and computes a graph G' on vertex set $Y' \cup Z$ such that $G'[Z] = G[Z]$ and a demand function $\phi': Y' \rightarrow \{0, \dots, d\}$ such that $\text{Sig}(G[Y \cup Z], Z, \phi|_Y) = \text{Sig}(G', Z, \phi')$ and $|D(G', \phi')| \leq d^3$. Moreover, the resulting G' and ϕ' are encoded using at most $f(d)$ bits and G' has at most $f(d)$ vertices, for some computable function f depending only on d .

Proof. The algorithm is as follows. First, compute $\text{Sig}(G[Y \cup Z], Z, \phi|_Y)$ in polynomial time using Lemmas 12 and 13. Then, generate all graphs G' with $G'[Z] = G[Z]$ in the order of increasing number of vertices (break the remaining

ties arbitrarily). For each graph G' , iterate over all possible $\phi': V(G') \setminus Z \rightarrow \{0, \dots, d\}$ and check whether $\text{Sig}(G[Y \cup Z], Z, \phi|_Y) = \text{Sig}(G', Z, \phi')$ as well as $|D(G', \phi')| \leq d^3$. Clearly, this procedure terminates and finds the required tuple (G', ϕ') , because $(G[Y \cup Z], \phi|_Y)$ witnesses its existence.

Without loss of generality, we may assume $Z = \{1, \dots, |Z|\}$. Now note that both requirements and facilities contain only set systems over Z , tuples of elements of Z , numbers in $\{1, \dots, \mathcal{O}(d^3)\}$, and the number of these entities is bounded by a function of d . Hence, $\text{Sig}(G[Y \cup Z], Z, \phi|_Y)$ can be encoded using at most $g(|Z|) \leq g(d)$ bits, where g is some monotone ascending, computable function. Thus, since $\text{Sig}(G[Y \cup Z], Z, \phi|_Y)$ is the only input to the procedure that finds G' and ϕ' , the procedure terminates after at most $f'(g(d))$ steps for some computable function f' , meaning that (G', ϕ') is of size at most $f'(g(d))$. \square

Now we can make the notion of replacing $G[Y]$ more precise; it involves an operation commonly referred to as glueing of graphs.

Definition 9 (Glueing). *Let G_1, G_2 be two graphs, both containing Z as a subset of their vertices. Glueing G_1 and G_2 on Z results in a graph denoted by $G_1 \oplus_Z G_2 := (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2))$, where $V(G_1)$ and $V(G_2)$ are treated as being disjoint except for Z .*

Below we only glue on Z for some vertex set Z defined in the context, so we will omit the index Z in the \oplus operation.

We arrive at the reduction rule aiming at reducing the size of Y .

Reduction Rule 4. *Let (G, ϕ, k) be an instance of VECTOR d -CONNECTIVITY(k) that is reduced with respect to Reduction Rules 1 and 2. Let $Y \in \mathcal{Y}$, where \mathcal{Y} is as in Definition 2, and let $Z = N_G(Y)$. Furthermore, let (G', ϕ') be as in Lemma 14. If $|Y| > |V(G') \setminus Z|$, then replace G by $G' \oplus (G - Y)$ and replace ϕ by $\phi|_{V(G) \setminus Y} \cup \phi'$.*

Before proving that Rule 4 is safe, we need a technical lemma that shows how paths from a demand vertex to a vertex-connectivity set are split over a separator.

Lemma 15. *Let G be a graph, (T, U) a separation of G , $Z = T \cap U$, $S \subseteq V(G)$, $v \in V(G) \setminus S$, and $d \in \mathbb{N}$. There are d v -independent paths from v to S in G if and only if there is an integer $i \in \{0, \dots, d\}$ and a partition of $Z \setminus \{v\}$ into four vertex sets A, B, C, D such that*

1. *if $v \in T \setminus U$ then $i = d$, and if $v \in U \setminus T$ then $i = 0$,*
2. *there is a $(v, i, A, B \cup (S \setminus U))$ -constrained path packing in $G[T \setminus C]$, and*
3. *there is a $(v, d - i, B, A \cup (S \cap U))$ -constrained path packing in $G[U \setminus D]$.*

Proof. (\Rightarrow): Assume first that there are d v -independent paths from v to S in G and let \mathcal{P} be a corresponding path packing (with overlap only in start vertex v). We may safely assume that paths in \mathcal{P} have no vertices of S as internal

vertices; else they could be shortened. We will select $A, B, C, D \subseteq Z \setminus \{v\}$ and $i \in \{0, \dots, d\}$ such that the path packings exist as stated in the lemma. For the purpose of getting a clear partitioning of the edges contained in Z , we show that one of the packings exists in $G[T \setminus C] - E(G[Z])$ and one of them in the remainder of the graph. Let us shorthand H for $G[T \setminus C] - E(G[Z])$.

Consider all paths in \mathcal{P} as being directed from v towards S , and consider the set \mathcal{P}_H of maximal, directed subpaths in H of paths in \mathcal{P} such that each path in \mathcal{P}_H contains at least one arc. Denote by \vec{H} the directed subgraph of H induced by \mathcal{P}_H . That is, \vec{H} contains precisely the vertices and arcs also contained in the paths in \mathcal{P}_H . We can now pick the sets $A, B, C, D \subseteq Z \setminus \{v\}$. The source vertices in \vec{H} not equal to v form the set A . Note that all source vertices except possibly v are contained in Z as each vertex on a path in \mathcal{P}_H but not in $Z \cup \{v\}$ must have a predecessor. Similarly, sink vertices in \vec{H} are contained in $Z \cup S$; we put those sink vertices that are contained in $Z \setminus \{v\}$ into B . Note that, as each path in \mathcal{P}_H has length at least one, there are no vertices of in- and outdegree zero and hence $A \cap B = \emptyset$. Vertices in Z that are used by paths in \mathcal{P}_H , but that are neither sources nor sinks, are put into D . Vertices of $Z \setminus \{v\}$ that are not on any path in \mathcal{P}_H are put into C . Clearly, A, B, C, D is a partition of $Z \setminus \{v\}$. Finally, we define $i = d$ if $v \in T \setminus U$, $i = 0$ if $v \in U \setminus T$, and if $v \in Z$, then i is defined as the outdegree of v in \vec{H} . The condition on d and i in the lemma is clearly fulfilled. We claim that \mathcal{P}_H is the desired path packing in $G[T \setminus C]$.

Showing that \mathcal{P}_H is a $(v, i, A, B \cup (S \setminus U))$ -constrained path packing in $G[T \setminus C]$: Clearly, H is a subgraph of $G[T \setminus C]$ and hence \mathcal{P}_H is contained in $G[T \setminus C]$. Observe that, since the paths in \mathcal{P}_H are vertex-disjoint (except for v) and each path has length at least one, sources and sinks in \vec{H} correspond to endpoints of these paths. Combining this with our observation from above that sources and sinks in \vec{H} are in $A \cup \{v\}$ and $B \cup S$, respectively, we infer that each path in \mathcal{P}_H starts in either v or A , and ends in $B \cup (S \setminus U)$. By definition of A , each vertex $w \in A$ has a path in \mathcal{P}_H starting in w and, furthermore, by the definition of i , there are i paths in \mathcal{P}_H that start in v . Hence, \mathcal{P}_H witnesses that there are $|A| + i$ paths from $A \cup \{v\}$ to $B \cup (S \setminus U)$ in H ; moreover, these paths do not touch C by definition. As the paths in \mathcal{P} are v -independent, so are the paths in \mathcal{P}_H . Hence, \mathcal{P}_H is a $(v, i, A, B \cup (S \setminus U))$ -constrained path packing in $G[T \setminus C]$.

Showing existence of a $(v, d - i, B, A \cup (S \cap U))$ -constrained path packing in $G[U \setminus D]$: Take the path packing \mathcal{P} and define a path packing \mathcal{P}' that contains all maximal subpaths of \mathcal{P} in $U \setminus D$. Note that \mathcal{P}' may contain paths of length zero. We consider also \mathcal{P}' as a set of directed paths, each arc inheriting its direction from \mathcal{P} . Clearly, \mathcal{P}' is contained in $G[U \setminus D]$. We claim that \mathcal{P}' is also $(v, d - i, B, A \cup (S \cap U))$ -constrained.

Consider the directed subgraph \vec{G}' of $G[U]$ induced by \mathcal{P}' . Let us find the endpoints of the paths in \mathcal{P}' . Clearly, if $v \in U \setminus D$, then v is such an endpoint. For the remaining endpoints, first, consider a path of length zero, represented by a vertex $w \neq v$. Since each path in \mathcal{P} has length at least one and since $w \neq v$, w has a predecessor u on a path in \mathcal{P} . Since w represents a path of length zero, $u \in (T \setminus U) \cup D$. By definition of D (since H does not contain any edges in Z),

each successor of a vertex in D on a path of \mathcal{P} is contained in $T \setminus U$. Hence, in fact $u \in T \setminus U$. The only vertices in U that have neighbors in $T \setminus U$ are contained in $T \cap U = Z$. This implies that $w \in Z$ and hence $w \in Z \setminus C$. Since \mathcal{P}' has empty intersection with D , we moreover have $w \notin D$. Hence, only two possibilities remain: $w \in A$ and $w \in B$. Assume that $w \in A$. Since the vertices in A are sources of \vec{H} , this implies that there is a path starting in $w \neq v$ in \mathcal{P} , a contradiction. It follows that $w \in B$. Since B represents sinks in \vec{H} , we moreover have $w \in S$ as, otherwise, there would be a path in \mathcal{P} ending in a vertex not contained in S . Thus, as also $w \in U$, each path of length zero in \mathcal{P}' ends in $S \cap U$ (and starts in B).

Next, consider paths of length at least one in \mathcal{P}' . Since they are pairwise vertex-disjoint (except for v), their endpoints correspond to the sources and sinks in \vec{G}' . Let w be a sink in \vec{G}' that is not contained in S . Since w is not in S , it has a successor x on a path in \mathcal{P} . As above, by the definition of D , each predecessor of a vertex in D on a path in \mathcal{P} is contained in $T \setminus U$. Hence, in fact $x \in T \setminus U$. The only vertices in U that have neighbors in $T \setminus U$ are contained in $T \cap U = Z$. Hence, we have $w \in Z$. Observe that $w \neq v$ as, otherwise, \mathcal{P} contains a cycle. The paths in \mathcal{P} are vertex-disjoint, thus, w does not have any incoming arcs in \vec{H} , meaning that it is a source in \vec{H} . This implies $w \in A$ by definition of A . Thus we obtain that \mathcal{P}' is a packing of paths, each of which ends in $A \cup (S \cap U)$.

It remains to prove that \mathcal{P}' contains $|B| + d - i$ paths that start in $\{v\} \cup B$; their v -independence is implied by the fact that these paths are subpaths of paths in \mathcal{P} . We claim that there are $d - i$ paths in \mathcal{P}' starting in v . First, if $v \notin U$ then $i = d$ by definition; hence, the claim is trivially true. If $v \in U \setminus T$ then, $i = 0$ and, clearly, each path in \mathcal{P} that starts in v induces one such path in \mathcal{P}' . Thus, the claim holds also in this case. Finally, if $v \in Z$, then i is the outdegree of v in \vec{H} . Recall that H does not contain any edge in Z . Hence, also in the final case there are $d - i$ paths in \mathcal{P}' that start in v .

To find the remaining $|B|$ paths, consider a vertex $w \in B$. Note that $w \neq v$ because $v \notin B$. By definition, w is a sink in \vec{H} and, since it is a part of a path in \mathcal{P} reaching S , it either is contained in S or has a successor on \mathcal{P} which is not contained in $T \setminus C$. In the first case, w represents a length-zero path starting in B in \mathcal{P}' . In the second case, w is a source in \vec{G}' by the vertex-disjointness of the paths in \mathcal{P} . Since the choice of w is arbitrary, and since the paths in \mathcal{P} are vertex-disjoint, each vertex in $B \setminus S$ is a source in \vec{G}' and hence has a path in \mathcal{P}' starting in this vertex. Thus, overall, there are $|B| + d - i$ paths from $\{v\} \cup B$ to $A \cup (S \cap U)$ in $G[U \setminus D]$ which are v -independent, as required. This completes the “if” part of the lemma.

(\Leftarrow): Assume that there is a partition A, B, C, D of $Z \setminus \{v\}$ and $i \in \mathbb{N}$ as described in the lemma and fix a $(v, i, A, B \cup (S \setminus U))$ -constrained path packing \mathcal{P}_T in $G[T \setminus C]$ and a $(v, d - i, B, A \cup (S \cap U))$ -constrained path packing \mathcal{P}_U in $G[U \setminus D]$. Consider the paths in \mathcal{P}_T as directed from $\{v\} \cup A$ to $B \cup (S \setminus U)$ and the paths in \mathcal{P}_U as directed from $\{v\} \cup B$ to $A \cup (S \cap U)$. Let us show that there is a packing of d v -independent paths from v to S in G .

Observe that \mathcal{P}_T and \mathcal{P}_U may overlap only in $A \cup B \cup (\{v\} \cap Z)$, as the graphs they are contained in overlap precisely in this vertex set. Consider the directed graph induced by the union of \mathcal{P}_T and \mathcal{P}_U . Denote by K the (weakly) connected component of this graph that contains v . By definition of \mathcal{P}_T and \mathcal{P}_U , vertex v is a source vertex. Let us first show that v has outdegree d in K . Otherwise, v must have a successor w in either A or B in both \mathcal{P}_T and \mathcal{P}_U . However, as the paths in \mathcal{P}_T start in A and the paths in \mathcal{P}_U start in B in both cases we get a contradiction. Thus, v is a source with precisely d outgoing arcs in K .

We claim that v is the only source in K . To see this, we first derive in- and outdegrees of all vertices other than v in K . Clearly, each vertex in $K - (A \cup B \cup \{v\})$ is either in S —and has indegree one and outdegree zero in this case—or has in- and outdegree exactly one. We claim that each vertex in $A \cup B$ has indegree at most one in K . This is clear for vertices in A , as only \mathcal{P}_U sends paths to A . Both packings \mathcal{P}_T and \mathcal{P}_U may send paths to a vertex $w \in B$ in the case that $w \in S$. Then, however, w is in a path of length zero in \mathcal{P}_U ,⁴ implying that indeed each vertex in $A \cup B$ has indegree at most one in K . Now for the sake of contradiction assume that there are two sources in K and consider a path in the underlying undirected graph of K between these two sources. On this path, there is a vertex with indegree at least two; a contradiction. Thus, v is the only source in K .

It now suffices to show that each vertex in $A \cup B$ either has in- and outdegree one in K or is a sink contained in S . As we have derived the same for the vertices in $V(K) \setminus (A \cup B \cup \{v\})$ above, and since the sum of all indegrees equals the sum of all outdegrees in K , this then implies that there are d vertex disjoint paths from v to S . Let thus prove that, indeed, each vertex in $A \cup B$ has either in- and outdegree one in K or is a sink contained in S .

We have shown above that each vertex in $A \cup B$ has indegree at most one in K . Since K has v as its only source, each of the vertices in $A \cup B$ has also indegree at least one in K . Since only \mathcal{P}_T has paths starting in A and only \mathcal{P}_U has paths starting in B , the outdegree of the vertices in $A \cup B$ is at most one in K . Now consider a sink $w \in V(K) \cap (A \cup B)$. Recall that \mathcal{P}_T contains a path starting in each vertex of A . Since $A \cap (B \cup (S \setminus U)) = \emptyset$, each of these paths has length at least one. Hence, $w \in B$. Since also \mathcal{P}_U has a path starting in w , it must be of length zero and thus $w \in S$ because the paths in \mathcal{P}_U end in $B \cup (S \cap U)$ and $A \cap B = \emptyset$. Thus we have shown that each vertex in K is either the source v with outdegree d , has in- and outdegree exactly one, or is a sink contained in S and has indegree exactly one. This means that there are d v -independent paths from v to S in G . \square

We are ready to show that Rule 4 respects YES and NO instances.

Lemma 16. *Rule 4 is safe.*

Proof. We claim that an even stronger statement holds. Namely, let G_1, G_2, \hat{G} be three graphs, each containing Z as a vertex subset such that $G_1[Z] = G_2[Z] =$

⁴ Recall that in a (v, i, A, B) -constrained path packing, we assume each path with endpoints in $A \cap B$ to be of length zero.

$\hat{G}[Z]$. Furthermore, let $\phi_1: V(G_1) \setminus Z \rightarrow \{0, \dots, d\}$, $\phi_2: V(G_2) \setminus Z \rightarrow \{0, \dots, d\}$, and $\hat{\phi}: V(\hat{G}) \rightarrow \{0, \dots, d\}$ be three demand functions, such that $|D(G_1, \phi_1)|, |D(G_2, \phi_2)| \leq d^3$ and such that $\text{Sig}(G_1, Z, \phi_1) = \text{Sig}(G_2, Z, \phi_2)$. Then $G_1 \oplus \hat{G}$ has a vector connectivity set of size k with respect to $\phi_1 \cup \hat{\phi}$ if and only if $G_2 \oplus \hat{G}$ has a vector connectivity set of size k with respect to $\phi_2 \cup \hat{\phi}$.

To see that our claim implies the lemma, set $G_1 := G[Y \cup Z]$, $G_2 := G'$, $\hat{G} := G - Y$ and define the demand functions ϕ_1, ϕ_2 and $\hat{\phi}$ accordingly. Then our claim implies that $G_1 \oplus \hat{G} = G[Y \cup Z] \oplus (G - Y) = G$ has a vector connectivity set of size k if and only if $G_2 \oplus \hat{G} = G' \oplus (G - Y)$ has such a set. That is, it implies that Rule 4 is safe.

Let us prove the claim. Note that, by swapping the names of G_1 and G_2 , as well as ϕ_1 and ϕ_2 , it suffices to prove one direction. Assume hence that $G_1 \oplus \hat{G}$ has a vector connectivity set S of size k with respect to $\phi_1 \cup \hat{\phi}$. Since $|D(G_1, \phi_1)| \leq d^3$ and $|N_{G_1 \oplus \hat{G}}(V(G_1) \setminus Z)| \leq |Z| \leq d$ we may apply Lemma 10 with $Y = V(G_1) \setminus Z$ and hence, we may assume that $S_1 := (V(G_1) \cap S) \setminus Z$ contains at most $d^3 + d$ elements. Thus, we have

$$(|S_1|, \text{Req}(G_1, Z, \phi_1, S_1), \text{Fac}(G_1, Z, S_1)) \in \text{Sig}(G_1, Z, \phi_1).$$

Since $\text{Sig}(G_1, Z, \phi_1) = \text{Sig}(G_2, Z, \phi_2)$, there is a set $S_2 \subseteq V(G_2) \setminus Z$ with $|S_1| = |S_2|$, $\text{Req}(G_1, Z, \phi_1, S_1) = \text{Req}(G_2, Z, \phi_2, S_2)$ and $\text{Fac}(G_1, Z, S_1) = \text{Fac}(G_2, Z, S_2)$. We claim that $S' := (S \setminus S_1) \cup S_2$ is a vector connectivity set for $G_2 \oplus \hat{G}$ with respect to $\phi_2 \cup \hat{\phi}$; clearly $|S| = |S'|$.

Let us check that this is true indeed. We consider vertices in $D(G_2, \phi_2)$, $D(\hat{G}[Z], \hat{\phi})$, and vertices in $D(\hat{G} - Z, \hat{\phi})$ individually. Let us start with $D(G_2, \phi_2)$. For each vertex $v_2 \in D(G_2, \phi_2)$, there is at least one vertex $v_1 \in D(G_1, \phi_1)$ with $\text{Sat}(G_1, Z, \phi_1, S_1, v_1) = \text{Sat}(G_2, Z, \phi_2(v_2), S_2, v_2)$. Let us show that the demand of v_2 is satisfied in $G_2 \oplus \hat{G}$ by S' .

Consider first the case that $v_1 \in S_1$. Then (A, B, C) is a satisfying connector for v_1 in G_1 for any three sets $A, B, C \subseteq Z$, mutually disjoint. In particular $(\emptyset, \emptyset, Z)$ is a satisfying connector for v_1 and since the set of satisfying connectors for v_1 equals the one for v_2 , $(\emptyset, \emptyset, Z)$ is a satisfying connector for v_2 . By the definition of satisfying connector either $v_2 \in S_2$, or there is a $(v_2, \phi_2(v_2), \emptyset, S_2)$ -constrained path packing in $G_2 - Z$, meaning that there are $\phi_2(v_2)$ v_2 -independent paths from v_2 to S_2 in $G_2 - Z$. Hence, the demand of v_2 is satisfied if $v_1 \in S_1$.

Now assume that $v_1 \notin S_1$ and observe that $(V(G_1), V(\hat{G}))$ is a separation of $\hat{G} \oplus G_1$. Since there are $\phi_1(v_1)$ v_1 -independent paths from v_1 to S in $\hat{G} \oplus G_1$, by Lemma 15 and since $v_1 \in V(G_1) \setminus V(\hat{G})$, there is a partition of Z into four sets A, B, C, D such that there is a $(v_1, \phi_1(v_1), A, B \cup (S \setminus V(\hat{G})))$ -constrained path packing \mathcal{P}_1 in $G_1 - C$ and a $(B, A \cup (S \cap V(\hat{G})))$ -constrained path packing $\hat{\mathcal{P}}$ in $\hat{G} - D$. Because $S \setminus V(\hat{G}) = S_1$, packing \mathcal{P}_1 is also $(v_1, \phi_1(v_1), A, B \cup S_1)$ -constrained. Thus \mathcal{P}_1 witnesses that $(A, B, C) \in \text{Sat}(G_1, Z, \phi_1(v_1), S_1, v_1)$, meaning that also $(A, B, C) \in \text{Sat}(G_2, Z, \phi_2(v_2), S_2, v_2)$. Applying the definition of satisfying connector again, we have a $(v_2, \phi_2(v_2), A, B \cup S_2)$ -constrained path

packing \mathcal{P}_2 in $G_2 - C$. Note that \mathcal{P}_2 is also $(v_2, \phi_2(v_2), A, B \cup (S' \setminus V(\hat{G}))$ -constrained. Applying again Lemma 15, the two path packings \mathcal{P}_2 and $\hat{\mathcal{P}}$ thus witness that there are $\phi(v_2)$ v_2 -independent paths from v to S in $\hat{G} \oplus G_2$.

Next, consider $v \in D(\hat{G} - Z, \hat{\phi})$; there are $\hat{\phi}(v)$ v -independent paths from v to S in $\hat{G} \oplus G_1$. Applying Lemma 15 with the separation $(V(G_1), V(\hat{G}))$, we obtain a partition of Z into A, B, C, D (since $v \in V(\hat{G}) \setminus V(G_1)$), a $(A, B \cup S_1)$ -constrained path packing \mathcal{P}_1 in $G_1 - C$ and a $(v, \hat{\phi}(v), B, A \cup (S \cap V(\hat{G})))$ -constrained path $\hat{\mathcal{P}}$ packing in $\hat{G} - D$. By the definition of provided connector, we have $(A, B, C) \in \text{Fac}(G_1, Z, S_1) = \text{Fac}(G_2, Z, S_2)$. Thus, again by the definition of provided connector, there is a $(A, B \cup S_2)$ -constrained path packing in $G_2 - C$. Applying again Lemma 15, the packings \mathcal{P}_2 and $\hat{\mathcal{P}}$ witness that there are $\hat{\phi}(v)$ v -independent paths from v to S' in $\hat{G} \oplus G_2$.

Finally, consider $v \in D(\hat{G}[Z], \hat{\phi})$; there are again $\hat{\phi}(v)$ v -independent paths from v to S in $\hat{G} \oplus G_1$. Now we apply Lemma 15 with the separation $(V(G_1), V(\hat{G}))$. This time, we obtain a partition of $Z \setminus \{v\}$ into A, B, C, D , and an integer i together with a $(v, i, A, B \cup S_1)$ -constrained path packing \mathcal{P}_1 in $G_1 - C$ and a $(v, \hat{\phi}(v) - i, B, A \cup (S \cap V(\hat{G})))$ -constrained path packing $\hat{\mathcal{P}}$ in $\hat{G} - D$. By the definition of provided special connector, the packing \mathcal{P}_1 witnesses that $(v, i, A, B, C) \in \text{Fac}(G_1, Z, S_1) = \text{Fac}(G_2, Z, S_2)$. Hence, again by this definition, there is also a $(v, i, A, B \cup S_2)$ -constrained path packing \mathcal{P}_2 in $G_2 - C$. Applying Lemma 15 again, the packings \mathcal{P}_2 and $\hat{\mathcal{P}}$ witness that there are $\hat{\phi}(v)$ v -independent paths from v to S' in $\hat{G} \oplus G_2$.

Overall we showed that each vertex v with nonzero demand $(\hat{\phi} \cup \phi_2)(v)$ has as many v -independent paths from v to S' in $\hat{G} \oplus G_2$, meaning that S' is a vector connectivity set. Since $|S'| = |S|$, this shows that Rule 4 is safe. \square

Putting things together. We can now state our kernelization procedure for instances (G, ϕ, k) of VECTOR d -CONNECTIVITY(k). The only missing piece is to argue why and how we may reduce vertices in G that are not contained in any set of $\mathcal{Y}(G, \phi, d)$.

Theorem 3. VECTOR d -CONNECTIVITY(k) has a vertex-linear polynomial kernelization.

Proof. Given an instance (G, ϕ, k) of VECTOR d -CONNECTIVITY(k) the kernelization proceeds as follows. Throughout, we refer to the current instance by (G, ϕ, k) and recall the use of $D = \{v \in V(G) \mid \phi(v) \geq 1\}$.

1. Apply Rule 1 exhaustively and then apply Rule 2 (this may return answer NO if we have more than $d^2 k$ demand vertices).
2. Apply Rule 4 once if possible.
3. Return to Step 1 if Rule 4 was applied.
4. Let $W := D \cup \bigcup_{Y \in \mathcal{Y}} N[Y]$. Perform the torso operation on W in G to obtain G' . That is, carry out the following steps:
 - (a) Start with $G' = G[W]$.
 - (b) For every pair $u, v \in W$, if there is a u, v -path in G with internal vertices from $V \setminus W$ then add the edge $\{u, v\}$ to G' .

5. Return (G', ϕ', k) as the kernelized instance, where $\phi' = \phi|_W$ is ϕ restricted to W .

Correctness. We already know that Rules 1 through 4 are correct; it remains to discuss the effect of the torso operation: Proposition 2 implies that minimal solutions S for (G, ϕ, k) are completely contained in the union of sets $X \in \mathcal{X}$, since only such vertices can contribute to S being a hitting set for \mathcal{X} . It follows, by Lemma 8, that every minimal solution S is also contained in W . Thus, if before the torso operation every vertex $v \in D$ has $\phi(v)$ paths to S then the same is true after the operation since there are shortcut edges for all paths with internal vertices from $V \setminus W$. The converse is more interesting.

Assume that (G, ϕ, k) is NO and fix an arbitrary set $S \subseteq W = V(G') \subseteq V(G)$ of size at most k ; we will show that S is not a solution for (G', ϕ', k) . By assumption S is not a solution for (G, ϕ, k) and, by Proposition 2, it is not a hitting set for $\mathcal{X} = \mathcal{X}(G, \phi)$. Accordingly, fix a set $X \in \mathcal{X}$ with $S \cap X = \emptyset$. By definition of \mathcal{X} , let $v \in X$ with $\phi(v) > |N(X)|$. By Lemma 8, there is a $Y \in \mathcal{Y}(G, \phi, d)$ with $X \subseteq Y$. Thus, $N[X] \subseteq N[Y] \subseteq W$. If there were $\phi(v)$ paths from v to S in G' then at least one of them avoids $N(X)$, since $|N(X)| < \phi(v)$; let P' denote a path from v to S in G' that avoids $N(X)$. Undoing the torso operation, we get a walk P in G , with additional interval vertices from $V(G) \setminus W$. Since $(V(G) \setminus W) \cap N(X) = \emptyset$, this walk also avoids $N(X)$ and implies that X contains at least one vertex of S ; a contradiction to $S \cap X = \emptyset$. Thus, no $S \subseteq V(G')$ of size at most k is a solution for (G', ϕ', k) , implying that (G', ϕ', k) is NO, as required.

Size. The output graph G' has vertex set $W = D \cup \bigcup_{Y \in \mathcal{Y}} N[Y]$ where D and \mathcal{Y} correspond to a fully reduced instance (G, ϕ, k) . By Rule 2 we have $|D| \leq d^2 k$. By Lemma 9 the set \mathcal{Y} contains at most $2^{d^3+d} d^2 k$ sets, each of size bounded by $f(d)$ for some computable function depending only on d . By Definition 2 the neighborhood $N(Y)$ of each set Y has size at most d . It follows that G' has $\mathcal{O}(k)$ vertices, as claimed. Clearly, the total encoding size for an instance can be bounded by $\mathcal{O}(k^2)$ since d is a constant.

Runtime. By Lemma 2, Rule 1 can be applied exhaustively in polynomial time. Clearly, Rule 2 can be applied in polynomial time as it only checks the number of nonzero-demand vertices. Finding one application of Rule 4 can be done by iterating over $Y \in \mathcal{Y}$ and applying Lemma 14 to each Y until we find a replacement subgraph that is strictly smaller; in total this takes polynomial time. Furthermore, repeating these steps whenever Rule 4 has been applied gives only a polynomial factor because each time the instance shrinks by at least one vertex. Finally, it is easy to implement the torso operation in polynomial time. \square

7 Kernelization lower bound

In this section, we prove that $\text{VECTOR CONNECTIVITY}(k)$ admits no polynomial kernelization unless $\text{NP} \subseteq \text{coNP/poly}$. We give a reduction from HITTING

$\text{SET}(m)$, i.e., HITTING SET with parameter number of sets, which also makes a polynomial Turing kernelization unlikely (cf. [7]). Since demands greater than $k + 1$ can be safely replaced by demand $k + 1$, implying $d \leq k + 1$, the lower bound applies also to parameterization by $d + k$.

Theorem 4. $\text{VECTOR CONNECTIVITY}(k)$ does not admit a polynomial kernelization unless $\text{NP} \subseteq \text{coNP/poly}$ and the polynomial hierarchy collapses.

Proof. We give a polynomial parameter transformation from $\text{HITTING SET}(m)$ to $\text{VECTOR CONNECTIVITY}(k)$, which is known to imply the claimed lower bound (cf. [1]). Let (U, \mathcal{F}, k) be an instance of $\text{HITTING SET}(m)$ with parameter $m = |\mathcal{F}|$; w.l.o.g. $k \leq m$. Let $n := |U|$. We construct a graph G on $2(k+1)m+n$ vertices that has a vector connectivity set of size at most $k' = (k+1)m+k = \mathcal{O}(m^2)$ if and only if (U, \mathcal{F}, k) is YES for $\text{HITTING SET}(m)$.

Construction. Make one vertex x_u for each element $u \in U$, and make $2(k+1)$ vertices $y_{1,F}, \dots, y_{k+1,F}, y'_{1,F}, \dots, y'_{k+1,F}$ for each set $F \in \mathcal{F}$. We add the following edges:

1. Add $\{y_{i,F}, y'_{i,F}\}$ for all $i \in \{1, \dots, k+1\}$ and $F \in \mathcal{F}$.
2. Add $\{x_u, y_{i,F}\}$ for all $i \in \{1, \dots, k+1\}$, $F \in \mathcal{F}$, and $u \in F$.
3. Make the set of all vertices $y_{i,F}$ a clique (not including any y' -vertex).

Set the demand ϕ of each $y'_{i,F}$ vertex to 2 and of each $y_{i,F}$ vertex to $(k+1)m+1$; all x -vertices have demand zero. Set the budget k' to $(k+1)m+k$. This completes the construction of an instance (G, ϕ, k') , which can be easily performed in polynomial time.

Correctness. Assume first that (G, ϕ, k') is YES and let S a vector connectivity set of size at most k' . Note that S must contain all vertices $y'_{i,F}$ since they have demand of 2 but only one neighbor (namely $y_{i,F}$). This accounts for $(k+1)m$ vertices in S ; there are at most k further vertices in S . Let T contain exactly those elements $u \in U$ such that $x_u \in S$; thus $|T| \leq k$. We claim that T is a hitting set for \mathcal{F} . Let $F \in \mathcal{F}$ and assume that $T \cap F = \emptyset$. It follows that S contains no vertex x_u with $u \in F$. Since at most k vertices in S are not y' -vertices, we can choose $i \in \{1, \dots, k+1\}$ such that S does not contain $y_{i,F}$. Consider the set C consisting of all y -vertices other than $y_{i,F}$ as well as the vertex $y'_{i,F}$. In $G - C$ we find a connected component containing $y_{i,F}$ and all x_u with $u \in F$ but no further vertices. Crucially, all other neighbors of $y_{i,F}$ are $y'_{i,F}$ and all y -vertices, and x -vertices only have y -vertices as neighbors. By assumption S contains no vertex of this connected component. This yields a contradiction cause C is of size $(k+1)m$ and separates $y_{i,F}$ from S , but since S is a solution with $y_{i,F} \notin S$ there should be $(k+1)m+1$ disjoint paths from $y_{i,F}$ to S . Thus, S must contain some x_u with $u \in F$, and then $T \cap F \neq \emptyset$.

Now, assume that (U, \mathcal{F}, k) is YES for $\text{HITTING SET}(m)$ and let T a hitting set of size at most k for \mathcal{F} . We create a vector connectivity set S by selecting all x_u with $u \in T$ as well as all y' -vertices; thus $|S| \leq k' = (k+1)m+k$. Clearly, this satisfies all y' -vertices. Consider any vertex $y_{i,F}$ and recall that its demand is $\phi(y_{i,F}) = (k+1)m+1$. We know that S contains at least one vertex x_u with

$u \in F$ that is adjacent to $y_{i,F}$. Thus, we can find the required $(k+1)m+1$ disjoint paths from $y_{i,F}$ to S :

- We have one path $(y_{i,F}, y'_{i,F})$ and one path $(y_{i,F}, x_u)$.
- For all $(j, F') \neq (i, F)$ we get one path $(y_{i,F}, y_{j,F}, y'_{j,F})$; we get $(k+1)m-1$ paths total.

It follows that (G, ϕ, k') is YES for VECTOR CONNECTIVITY(k).

We have given a polynomial parameter transformation from HITTING SET(m), which is known not to admit a polynomial kernelization unless $\text{NP} \subseteq \text{coNP/poly}$ [4] (see also [7]). This is known to imply the same lower bound for VECTOR CONNECTIVITY(k) [1]. \square

8 Conclusion

We have presented kernelization and approximation results for VECTOR CONNECTIVITY and VECTOR d -CONNECTIVITY. An important ingredient of our results is a reduction rule that reduces the number of vertices with nonzero demand to at most $d^2 \text{opt}$ (or, similarly, to at most $\text{opt}^3 + \text{opt}$ or $k^3 + k$). From this, one directly gets approximation algorithms with ratios d^2 and $(\text{opt}^2 + 1)$; we improved these to factors d and opt , respectively, by a local-ratio type algorithm. Recall that VECTOR d -CONNECTIVITY is APX-hard already for $d = 4$ [3].

On the kernelization side we show that VECTOR CONNECTIVITY(k) does not admit a polynomial kernelization unless $\text{NP} \subseteq \text{coNP/poly}$. Since demands greater than $k+1$ can be safely replaced by demand $k+1$ (because they cannot be fulfilled without putting the vertex into the solution) the lower bound extends also to parameter $k+d$. For VECTOR d -CONNECTIVITY(k), where d is a problem-specific constant, we give an explicit vertex-linear kernelization with at most $f(d) \cdot k = \mathcal{O}(k)$ vertices; the computable function $f(d)$ is superpolynomial in d , which is necessary (unless $\text{NP} \subseteq \text{coNP/poly}$) due to the lower bound for $d+k$.

Finally, the reduction to k^3+k nonzero demand vertices allows an alternative proof for fixed-parameter tractability: We give a randomized FPT-algorithm for VECTOR CONNECTIVITY(k) that finds a solution by seeking a set of size k that is simultaneously independent in each of $\ell = k^3 + k$ linear matroids, each of which handles one demand vertex; for this we use an algorithm of Marx [11] for linear matroid intersection, which is fixed-parameter tractable in $k + \ell = \mathcal{O}(k^3)$.

Acknowledgments. The authors are grateful to anonymous reviewers for helpful comments improving the presentation and technical content of the paper. In particular, a reviewer commented on the definition of signatures, leading to a significantly simpler way of computing them; other reviewers pointed out that a non-constructive kernelization can be obtained from the literature on meta kernelization (in particular, from Fomin et al. [5]).

Stefan Kratsch was supported by the German Research Foundation (DFG), KR 4286/1. Manuel Sorge was also supported by the German Research Foundation (DFG), project DAPA, NI 369/12.

References

1. H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Kernelization lower bounds by cross-composition. *SIAM J. Discrete Math.*, 28(1):277–305, 2014.
2. E. Boros, P. Hegghernes, P. van ’t Hof, and M. Milanic. Vector connectivity in graphs. *Networks*, 63(4):277–285, 2014.
3. F. Cicalese, M. Milanic, and R. Rizzi. On the complexity of the vector connectivity problem. *Theoretical Computer Science*, 2015. Available online.
4. M. Dom, D. Lokshtanov, and S. Saurabh. Incompressibility through colors and IDs. In S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. E. Nikolettseas, and W. Thomas, editors, *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5555 of *Lecture Notes in Computer Science*, pages 378–389. Springer, 2009.
5. F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Linear kernels for (connected) dominating set on graphs with excluded topological subgraphs. In N. Portier and T. Wilke, editors, *30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013, February 27 - March 2, 2013, Kiel, Germany*, volume 20 of *LIPICs*, pages 92–103. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.
6. H. W. Hamacher and Z. Drezner. *Facility location: applications and theory*. Springer, 2002.
7. D. Hermelin, S. Kratsch, K. Soltys, M. Wahlström, and X. Wu. A completeness theory for polynomial (Turing) kernelization. In G. Gutin and S. Szeider, editors, *Proceedings of the 8th International Symposium on Parameterized and Exact Computation (IPEC)*, volume 8246 of *Lecture Notes in Computer Science*, pages 202–215. Springer, 2013.
8. S. Jukna. *Extremal combinatorics - with applications in computer science*. Texts in theoretical computer science. Springer, 2001.
9. S. Kratsch and M. Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 450–459. IEEE Computer Society, 2012.
10. D. Lokshtanov. Private communication (unpublished), 2014.
11. D. Marx. A parameterized view on matroid optimization problems. *Theor. Comput. Sci.*, 410(44):4471–4479, 2009.
12. J. Oxley. *Matroid Theory*. Oxford University Press, 2011.